ニューラルネットワークの学習による
フィードバック制御系の構築に関する研究

2001年

中西 弘明

目 次

第 1 章

序論

1.1 ニューラルネットワークとその構造

ニューラルネットワークとは，脳などにおける神経回路網のことである．一つ一つの神経細胞は単純であるにもかかわらず，それらが多数集まった集合体である脳は，学習や記憶をはじめとして複雑かつ多彩な機能をもつために，従来より脳のモデルを作成し，研究することが行なわれてきた．脳のモデルを作成し研究を行なう目的には大別すると二つ存在する．一つは脳の仕組みを解明することであり，もう一つは優れた情報処理機能を持つシステムを構築することである．本研究では後者の立場をとり，ニューラルネットワークとは神経回路網をモデルにした人工システムのことを指すものとする．簡単な処理を行なう処理ユニットにより神経細胞をモデル化し，それらが互いに結合することによって神経回路網のモデルであるニューラルネットワークを構築することができる．処理ユニットの結合法からみると，ニューラルネットワークは二つに区別できる．処理ユニット間にはフィードバック結合を含まず，情報の伝播方向が一定に定まっている階層構造型ニューラルネットワークと，ユニットが相互に結合している相互結合型ニューラルネットワークである．それぞれの型のニューラルネットワークは優れた情報処理能力を有していることは，多くの研究の結果により指摘されている．特に，階層構造型ネットワークは関数の近似能力を持ち，その能力はパターン認識を初めとして様々な応用に有効である．また，相互結合型ネットワークは組合せ最適化問題の発見的解法の一つとして有効であることが知られている．図 1.1および，図 1.2にそれぞれのネットワークの例を示す．本研究において，階層構造型ニューラルネットワークが基本的に用いられる．よって，ここでは図 1.2のような中間層が 1 層である階層構造型ニューラルネットワークの入出力関係を簡単に説明する．階層構造型ニューラルネットワークでは，各ユニットをそれぞれの入力源および出力先により分類することができる．入力源から分類すると，ネットワークの外部から入力を受けるものとネットワークに含ま

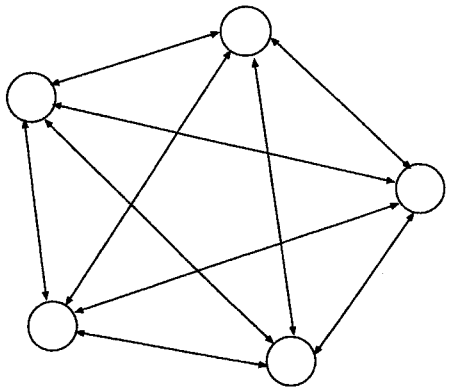


図 1.1: Recurrent neural network

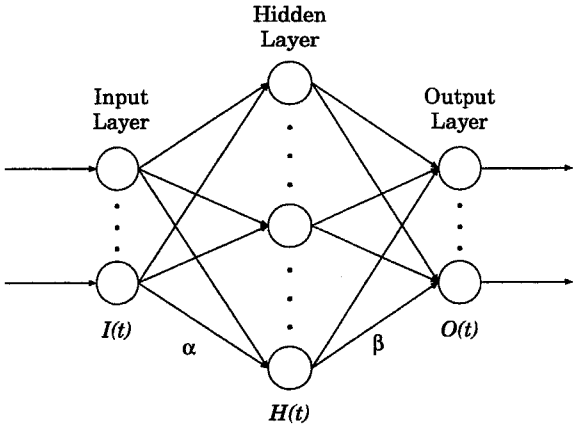


図 1.2: Three layered neural network

れている処理ユニットからのみ入力を受けるものに分けられる．出力先で分類すると，ネットワークの外部に出力を与えるものと外部には出力を与えないものとに分けられる．これら分類に基づいて，外部から入力を受けるユニットを入力ユニット，外部に出力を与えるユニットを出力ユニット，外部と入出力関係の存在しない中間ユニットと呼ぶ．中間ユニットは外部から見えないので，隠れユニットとも呼ばれる．階層構造型ニューラルネットワークでは，図1.2のようにユニットが完全に階層を形成して結合している．入力ユニット，出力ユニット，中間ユニットそれぞれのみからなる層をそれぞれ入力層，出力層，中間層と呼ぶ．中間層は隠れ層と呼ばれることもあり，複数の中間層をもつネットワークも考えることができる．階層構造型ネットワークでは入力から出力の方向に層間結合のみが存在し，それぞれの層では層内結合が存在しない．各ユニットは直前の層に含まれるユニットの出力を結合度と呼ばれる重みをつけて受け，その値によって出力を決定する．神経回路の細胞にならってこのことを活性化といい，各ユニットの出力のことを活性値という．活性化を表す関数，すなわちユニットにおける入出力間の関数を活性化関数という．また，神経細胞にならって，ユニットの活性値がある定められた値を越えたとき，そのユニットは活性化した，あるいはそのユニットは発火したという．入力層のユニット i が外部からの入力 $u_i(t)$ を受けて活性化関数 F_I で活性化するものとすると，その活性値 $I_i(t)$ は

$$I_i(t) \;=\; F_I(u_i(t)) \tag{1.1}$$

で決まる．入力層と中間層の間の結合度をα_{ij}，i 番目のユニットのしきい値をθ_i，中間層のユニットの活性化関数を F_Hとすると，中間層に含まれる i 番目のユニットの活性値 $H_i(t)$ は

$$net_i^H(t) \;=\; \sum_{j=1}^{m} \alpha_{ij} I_j(t) + \theta_i \tag{1.2}$$

$$H_i(t) \;=\; F_H(net_i^H(t)) \tag{1.3}$$

のように決定する．中間層と出力層の結合度をβ_{ij}，出力層のユニットの活性化関数をF_Oとすると，出力層のi番目のユニットの活性値$O_i(t)$は

$$net_i^O(t) = \sum_{j=1}^{N} \beta_{ij} H_j(t) \tag{1.4}$$

$$O_i(t) = F_O(net_i^O(t)) \tag{1.5}$$

のように決定する．$O_i(t)$が階層構造型ネットワークの出力$x_i(t)$となる．以上により，m入力n出力の階層構造ニューラルネットワークはm次元ベクトル$\boldsymbol{u}(t)$をn次元ベクトル$\boldsymbol{x}(t)$に写す静的な写像を表していることが分かる．階層構造型ニューラルネットワークに対して適当な結合度を与えると，任意の連続写像を任意の精度で近似できることが知られている [1]．この証明で用いられているように，階層構造型ニューラルネットワークの入力層及び出力層の活性化関数として恒等関数が，中間層の活性化関数としてはシグモイド関数がよく用いられる．シグモイド関数とは単調かつ有界な関数であり，双曲正接関数 tanh やロジスティック関数などが用いられることが多いが，これらの関数の微分係数はその関数値を用いて求めることができるために，微分を求めることが容易であるためである．このように，活性化関数としてシグモイド関数を用いる活性則をシグモイド型活性則と呼ぶ．この他に，中間層の活性関数にRBF(Radial Basis Function)と呼ばれる関数族を用いたRBF型活性則もよく用いられる [2]．RBFとは得られた観測データを中心に展開し，系を支配する関数や系を近似する関数を求めること目的として用いられる．RBFをそのままニューラルネットワークに応用すると，観測データ数と同じ中間ユニット数が必要となってしまうため，莫大な数の中間ユニット数となり現実的とはいえなくなることがある．このために，実際には観測データ数より少ない中間ユニット数のものを用いることが多い．また，RBF型の活性則では，入力と中心入力と呼ばれる各ユニットに埋め込まれている入力パターンとの距離よって発火するのため，RBFニューラルネットワークの中間ユニットはネットワークの入力のパターンに対して発火しているといえる．RBFにはいくつかの関数が含まれるが，ニューラルネットワークではよく用いられているものはガウシアンであるので，これに基づいて説明を行なう．ガウシアンRBFニューラルネットワークの中間ユニットでは，(1.6)のような活性則により活性値が決定される．

$$H_i = \exp\left(-\sum_j \frac{(I_j - \theta_{ij})^2}{\sigma_{ij}^2}\right) \tag{1.6}$$

ここで，σ_{ij}およびθ_{ij}はガウシアンの分散と平均であり，シグモイド型活性則における結合度および中間ユニットのしきい値に相当する．シグモイド型活性則とガウシアンRBF型活性則の間に存在する最も大きな違いは活性化の局所性である．ガウシアンRBF型活性則では，(1.6)に示されているように，その中間ユニットが対象とする入力パターン以外の入力では急激に発火しなくなるため，その中間ユニットはネットワークの入出力関係に局所的な影響しか及ぼすことはない．すなわち，一つの中間ユニットは特定の入力の

領域のみを受け持ち，異なる複数のユニットが集まることにより入力の領域全体が覆われる．これに対して，シグモイド型の活性則では，入力の重み付和に応じて，(1.2)，(1.3) のように活性化するので，全ての中間ユニットもネットワークの入出力関係の全領域に対する影響が必ず存在するため，結合度の変化の影響が及ぶ入力の範囲も大域的となる．このため，ある結合度が少し変化したとすると，その変化の影響が入出力の全域に及ぶ．このため，シグモイド型活性則では，次節で述べる学習によってある結合度が変化すると，その影響はネットワークの入出力関係全体に影響し，すでに正しく学習がすすんでいた部分に対して悪影響を及ぼす可能性がある．これに対して，RBF 型活性則では結合度の変化はネットワークの入出力関係に局所的にしか影響を及ぼさないので，すでに正しく学習がすすんでいる部分には影響を与えない．このために学習の効率が RBF 型活性則の方が高くなるが，近似の精度はシグモイド型活性則に劣る．このために目的に応じて活性則を選択する必要がある．

本研究では，階層構造型ニューラルネットワークの学習によりフィードバック制御系を構築する方法について考える．図 1.1のようにニューラルネットワークをフィードバック制御器として用いる場合には，制御対象の状態がニューラルネットワークに入力され，制御対象への入力をニューラルネットワークが出力する．このとき，ニューラルネットワークの出力はネットワーク外部で処理が加わった後でネットワークの入力となって戻ってくることは図 1.1より明らかである．このために，ニューラルネットワークの構造は階層構造型であったとしても，外部にフィードバック構造を含んでいるために，全体として考えると相互結合型ネットワークとなることが分かる．

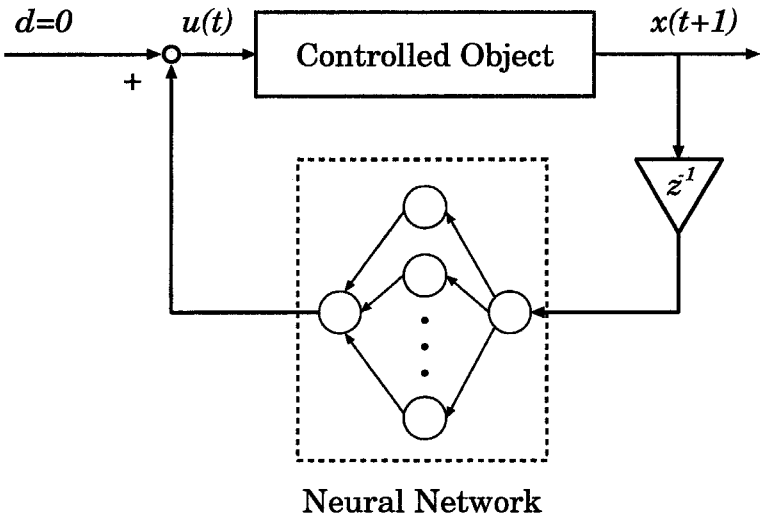


図 1.3: Typical block diagram of a control system including a neural network

1.2 学習

ニューラルネットワークの持つ最大の特徴は学習である．階層構造型ニューラルネットワークにおける学習とは，ニューラルネットワークにある望ましい写像を近似させるために結合度を変化させることをいう．学習を行う方法は，教師あり学習と教師なし学習の2つに分けることができる．教師あり学習とは，望ましい入出力データ (教師信号) が何らかの方法を用いて既に獲得されており，そのデータに基づいて学習を行なうことをいう．ニューラルネットワークにおける教師つき学習の例としては，Rumelhart による誤差逆伝播法がある [3]．誤差逆伝播法とは，教師信号と現在の結果との誤差二乗関数の結合度による勾配を求める際に，教師信号との現在の出力との誤差信号がネットワークにおけるデータ処理の流れを逆に伝播するように見えることからこのように呼ばれる．得られた勾配に沿った方向に結合度を変更することにより誤差二乗関数を小さくするように学習が進む．この方法は，非常に簡単であるにもかかわらず有効であるために，階層構造型ニューラルネットワークではもっとも広く用いられている方法である。しかし，教師信号が存在する場合であっても，相互結合型ネットワークでは勾配を求めることは，階層構造型ネットワークに比べると一般に困難である．

教師なし学習には相互結合型のニューラルネットワーク，特にホップフィールドモデルにより，組合せ最適化問題を解く際に用いられる学習法などがある．このような最適化問題では，何らかのエネルギーを考えて，そのエネルギーが減少するように結合度を変化させる．この他にも，教師なし学習としてはHebbの学習規則と呼ばれる学習もある．

ニューラルネットワークを制御に応用する研究は，システム同定においてニューラルネットワークを用いる研究と，制御系として用いる研究に分けることができる．システム同定においては制御対象を観測することにより教師信号を作成することができるために，教師あり学習を用いることができる．制御系設計においては，あらかじめ決められた望ましい制御系の応答を規範モデルから作成し，それに追従するように制御系を変化させる適応制御系への応用がまず考えられる．この方法において，規範モデルにより教師信号を作成することができるために，教師あり学習の適用が可能である．しかし，最適制御やロバスト制御系を構築する場合には，事前に解が求まっていない限り教師信号を得ることができないために，教師なし学習を行う必要がある．

1.2.1 誤差逆伝播法

本節では，本論文で提案する勾配法に基づく学習法と比較するために，ニューラルネットワークの学習方法として最も知られている誤差逆伝播法について簡単に述べる．ここでは図1.2のような三層構造型ニューラルネットワークについて考え，簡単のために入力層及び出力層の活性化関数を恒等関数，中間層の活性化

関数をシグモイド関数Fとする．ここで，因果関係より (1.7)，(1.8)，(1.9) が成立する．

$$\frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{\beta}} = \frac{\partial \boldsymbol{\beta}}{\partial \boldsymbol{\alpha}} = \mathbf{0} \tag{1.7}$$

$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{\beta}} = 0 \tag{1.8}$$

$$\frac{\partial \boldsymbol{I}}{\partial \boldsymbol{\alpha}} = \frac{\partial \boldsymbol{I}}{\partial \boldsymbol{\beta}} = \mathbf{0} \tag{1.9}$$

ニューラルネットワークの学習における評価関数Jとして

$$J = J(\boldsymbol{O}) \tag{1.10}$$

とする．ここで任意の結合度をwで表すと，Jの勾配$\partial J/\partial w$は

$$\frac{\partial J}{\partial w} = \sum_i \frac{\partial J}{\partial O_i}\frac{\partial O_i}{\partial w} \tag{1.11}$$

であるので，Jの勾配の計算は$\partial O_i/\partial w$を求めることに帰着される．ここで，ネットワークの入出力関係が

$$O_i = \sum_j \beta_{ij} H_j \tag{1.12}$$

$$H_i = F_i(\sum_j \alpha_{ij} I_j) \tag{1.13}$$

であることと，(1.7)，(1.8)，(1.9) を考慮すると，(1.14)，(1.15)，(1.14) を得る．

$$\frac{\partial O_i}{\partial \beta_{pq}} = \frac{\partial}{\partial \beta_{pq}}\sum_j \beta_{ij} H_j = \sum_j \delta_{ip}\delta_{jq} H_j = \delta_{ip} H_q \tag{1.14}$$

$$\frac{\partial H_i}{\partial \alpha_{pq}} = F_i' \frac{\partial}{\partial \alpha_{pq}}\sum_j \alpha_{ij} I_j = F_i'\, \delta_{ip}\delta_{jq} I_j = F_i'\, \delta_{ip} I_q \tag{1.15}$$

$$\frac{\partial O_i}{\partial \alpha_{pq}} = \frac{\partial}{\partial \alpha_{pq}}\sum_j \beta_{ij} H_j = \sum_j \beta_{ij}\frac{\partial H_j}{\partial \beta_{pq}} = \sum_j \beta_{ij} F_j'\delta_{jp} I_q = \beta_{ip} F_p' I_q \tag{1.16}$$

さらに，(1.14)，(1.16) を (1.11) に代入することにより評価関数Jのネットワークの結合度に対する勾配を求めることができる．

$$\begin{aligned}\frac{\partial J}{\partial \beta_{pq}} &= \sum_i \frac{\partial J}{\partial O_i}\frac{\partial O_i}{\partial \beta_{pq}} \\ &= \sum_i \delta_{ip} H_q \frac{\partial J}{\partial O_i} \\ &= \frac{\partial J}{\partial O_p} H_q\end{aligned} \tag{1.17}$$

$$\begin{aligned}\frac{\partial J}{\partial a_{pq}} &= \sum_i \frac{\partial J}{\partial O_i}\frac{\partial O_i}{\partial \alpha_{pq}} \\ &= \sum_i \beta_{ip}\frac{\partial J}{\partial O_i} F_p' I_q\end{aligned} \tag{1.18}$$

さらに，ネットワークの望ましい出力として教師信号 d_i を考え，評価関数 J を

$$J = \sum_i \frac{1}{2}(O_i - d_i)^2 \tag{1.19}$$

とする．このとき

$$\frac{\partial J}{\partial O_i} = O_i - d_i \tag{1.20}$$

を誤差信号と呼ぶ．(1.17)，(1.18) を用いて評価関数 Jの勾配を求めるときに，誤差信号が出力層から入力層の方向へと逆に伝播するように見える．このようにして評価関数 J(1.19) の勾配を求める方法を誤差逆伝播法という．ここでは，簡単のためにしきい値を扱わなかったが，しきい値の学習も誤差逆伝播法で行なうことができる．

1.3 本研究の背景・他の研究との関連

ニューラルネットワークを制御に応用する際，逆問題を取り扱うことが多かった [4]．逆問題とは，あるシステムの入力と出力をそれぞれ u と yとして，

$$y = F(u) \tag{1.21}$$

のような入出力関係式が存在するものとすると，u を与えて yを決定する順問題と異なり，yを与えて u を決定する，すなわち，次の関係式を解く問題である．

$$u = F^{-1}(u) \tag{1.22}$$

制御対象の理想的な応答が与えられたときに，その応答となる制御入力を求めるという問題は逆問題の典型例である．Psaltis らによると，制御対象の入出力関係の逆写像を学習する機構として，Indirect learning, General learning, Specialized learning が挙げられている [5]．それぞれの機構においてニューラルネットワークと制御対象の配置が少し異なり，望ましい制御対象の出力，あるいは制御対象の出力をニューラルネットワークに入力し，制御対象への理想的な制御入力，あるいはそのときの制御入力を出力するように学習を行なう．理想的な制御入力あるいは実際の制御入力と，ニューラルネットワークの出力の差は実現された逆システムの誤差であるので，この誤差を減少させることができればニューラルネットワークは逆システムを学習することが可能である．Psaltis らにより提案された方法では，ニューラルネットワークの学習アルゴリズムとして 1.2.1節で述べた誤差逆伝播法を用いる．しかし，Psaltis らの方法にはそれぞれ問題点があることが指摘されており，実際には用いることは困難である [6]．このような困難を解決する方法として，川人によりフィードバック誤差学習法 [7] が提案された．フィードバック誤差学習法において，フィー

ドバック制御器による入力を誤差信号としてニューラルネットワークの学習を行い．学習アルゴリズムはやはり誤差逆伝播法を用いる．これらの方法は，学習アルゴリズムとして誤差逆伝播法を用いることを前提として導出されている．ニューラルネットワークを制御系に応用する際に，学習アルゴリズムとして誤差逆伝播法を用いることを考えると，教師信号，あるいは誤差信号が必要となる．このために，誤差逆伝播法を用いることに拘ると，ニューラルネットワークによる制御系設計は限られた対象や目的にしか用いることができなかった．ニューラルネットワークにより熟練者のプラントの操縦方法を学習を行ない，学習後のネットワークにより制御を行なう方法も研究されたが，熟練者の操縦パターンを教師データするには膨大なデータサンプリングして学習を行なわなければならない．また，Narendra はニューラルネットワークを適応制御系に応用したが，学習アルゴリズムとしてはやはり誤差逆伝搬法であった [8]．誤差逆伝播法は最急降下法であったので学習の速度が遅い．停留点付近での振動現象など好ましくない現象を持っているために，対象の急激な変化に対して速やかに適応が完了するとはいえなかった．しかし，ニューラルネットワークを用いた適応制御系に関しては，Calise らの最近の研究 [9,10] により大きく進歩し，学習の速度の問題点は克服されている．しかし，適応制御のみでは様々な制御仕様を満足することは困難であり，オンライン学習を行なうニューラルネットワークはオフライン学習で求めたニューラルネットワークなど固定された制御系の補助として用いるべきである．

また，得丸らは誤差逆伝播法によるオンライン学習により，最適制御系をニューラルネットワークを用いて構築する方法 [11] を提案しているが，評価関数を直接用いて学習していないので最適であるかどうかには疑問が残る．また，誤差逆伝播法を用いるために最適制御理論からの結果に近似を施したものを用いるために，さまざまな対象に対しても有効であるかどうかは明確ではなかった．これに対して Narendra や Widrow らにより時間を遡る誤差の逆伝播 [12,13] が考えられ，誤差逆伝播法を拡張することにより，最適制御問題を解くことが可能となった．この方法は中間層にのみ相関結合を許す混合構造型ネットワークによる最適制御のシンセシス [14] でも用いられており，相互結合型ニューラルネットワークの学習アルゴリズムとしても有効な方法である．しかし，誤差信号の時間を遡る伝播という形で，誤差逆伝播法と同じような学習アルゴリズムに拘っているといえる．また，最適制御入力時系列を求めることができるのみであったので，制御系の設計として用いるには不十分であった．本研究の第 2章にある勾配法に基づく学習法は，これらの学習法を考え直すことにより導かれたものであり，Narendra らの方法の一般化である．著者とは別に，平澤らも一般化学習ネットワーク [15] により，評価関数の任意の勾配を求める方法を提案し，制御系の構築にニューラルネットワークの学習を応用する研究を行なっている．しかし，評価関数の勾配を用いる学習法を用いて求めることが困難である制御系が多いことが，[16] により明らかとなった．勾配を計算しない学習方法として摂動法 [17] があり，これを制御系の構築に応用する試みもなされているが，学習の効率

が高いとはいえないことが問題となることが分かっている．さらに，ニューラルネットワークを用いた制御系ではロバスト性はほとんど考慮されておらず，定量化されたロバスト性やロバスト性解析はこれまで行なわれていない．ロバスト適応制御にニューラルネットワークを用いる研究はCaliseらにより行なわれているが，そのロバスト性に対する定量的な評価は行なわれていない．

1.4 本論文の構成

本論文の構成は以下の通りである．

第2章では，ニューラルネットワークにより最適フィードバック制御系を学習する方法について述べる．学習アルゴリズムとして，勾配法に基づく学習とPowellの共役方向法に基づく学習の二つを示し，それぞれの有効性を示す．特に，Powellの共役方向法に基づく学習では，制御対象の状態方程式が学習アルゴリズムに含まれないために，汎用性が高いことを示す．

第3章では，ニューラルネットワークの学習によりロバスト制御系を学習する方法について考察し，．制御対象に含まれる様々な種類の不確かさに対してロバスト制御系を構築する学習アルゴリズムを提案する．特に，学習後のロバスト性を定量的に評価することができる学習法について述べるとともに，任意の制御系のロバスト性を定量的に解析・評価を行なう際にも，ニューラルネットワークの学習を用いることが有効であることを示す．

第4章では，制御対象の情報を学習アルゴリズムに用いることができない場合の例として無人飛行機の自律飛行制御系の構築にニューラルネットワークの学習を適用した例を示す．Powellの共役方向法に基づく学習法を，フライトシミュレータに組み込むことにより無人機に関する詳細な知識を必要とせずに，有効な制御系を容易に構築することができることを示す．また，風などの確率的な外乱下において有効な制御系を構築するために，確率的な不確かさに対してロバストな制御系の構築方法をいくつか示し，それらの特性を明らかにする．

第5章を結論とし，本研究で得られた成果をまとめる．

第 2 章

学習による最適フィードバック制御系の構築

2.1 緒言

最適制御問題とは微分方程式，または差分方程式で表される制御対象に対して，ある与えられた評価関数を最小あるいは最大にする制御入力を決定する問題である．変分法や最大値原理によると，最適制御入力とは状態フィードバックの形で一般に実現できることが知られている．特に，制御対象が線形でかつ評価関数が二次形式であるような場合には解析解が存在する．しかし，一般の場合には解析的に解くことは非常に困難であるため，最適制御問題を解くということは一般的には数値的に解くことを意味する．しかし，数値的に得られる最適制御入力はフィードバック解でなくフィードフォワード解であるために，制御対象の初期条件に解が依存する上に，雑音に対する耐性がないなどの難点がある．このような難点を克服するためには，フィードバック制御系を構築する必要があるが，このためには制御対象の線形近似モデルを作成し，二次形式の評価関数を用いる必要がある．しかし，設計に用いたモデルと現実の制御対象の間に存在する誤差のために，最適な制御であるとはいえないだけでなく，設計仕様を満足しない場合もある．

階層構造型ニューラルネットワークは，任意の連続写像を任意の精度で学習することができる [1] という長所を持つ．このために，階層構造型ニューラルネットワークを制御工学に応用する試みが多くなされている [5,8]．ニューラルネットワークの学習により，非線形制御対象の同定が可能となり，正確な非線形モデルを作ることが可能となる．また，制御系設計では主として適応制御系に適用されているが [5,8]．これは Rumelhart らによる誤差逆伝播法 [3] を用いることができるからである．しかし，誤差逆伝播法を用いたニューラルネットワークの学習により最適制御系を構築するためには，最適な入力とそれに対応する最

適な状態を学習のためのデータとして求めねばならない．しかし，このためには最適制御問題を数値的に解く必要がある．混合構造型と呼ばれる中間層にのみ層内結合を許したニューラルネットワークを用いて，最適制御入力を決定する方法が研究されている [14]．だが，これは最適制御入力を時系列の形で求める，すなわちフィードフォワード解を求めることを目的としていた．この他に，最適制御問題にニューラルネットワークを応用したものに [11,18,19] などがあり，これらの研究では学習アルゴリズムとして誤差逆伝播法が用いられている．しかし，[11] では学習により最適化する評価関数は制御系の評価関数と異なるため，学習により最適制御系を実現できているかどうかが曖昧であった．また，[18] ではハミルトニアンの最適化を行なうが，制御器の役割を果たすニューラルネットワークがサンプリングタイム数だけ必要である．これは誤差逆伝搬法を導くために相互結合型となっていたネットワークを時間的に展開して階層構造型ネットワークへ変換を行うためである．このような変換は Rumelhalt により提案されており [3], [13,14] における最適制御入力を求める学習方法にも用いられている．しかし，このためには大規模なネットワークを利用した最適化問題を解く必要があった．このような難点は，学習アルゴリズムをとして誤差逆伝播法を用いることに固執したことが原因である．よって，ニューラルネットワークを用いて最適制御問題を効率よく解くには，新しい学習アルゴリズムが必要となる．また，最適制御入力の決定法では応用範囲が狭く，制御系の構築に有効とはいえず，実際に有用である最適フィードバック制御を求める方法が必要である．

本章ではニューラルネットワークを用いて最適フィードバック制御問題を解く学習方法を提案する．学習アルゴリズムとして勾配法に基づく学習アルゴリズムと Powell の共役方向法に基づく学習アルゴリズムの二つを示す．これらの学習アルゴリズムを応用することで様々な最適制御問題のフィードバック解が求められることを示す．

2.2 最適フィードバック制御系

2.2.1 最適制御問題

本章では，制御対象を時不変，集中定数，離散時間システムとする．制御対象の次数は既知であり，全ての状態変数が測定可能であるとする．最適制御問題とはこのようなシステムに対して与えられた評価関数を最大，または最小にするような制御入力を求める問題である [20–22]．しかし，この問題を解析的に解くことは一般的に困難であるため，最適解は数値計算により得られる [23]．しかし，数値計算によって最適入力は時系列の形しか得ることができない．．制御器の構造から見ると，数値解による制御はフィードフォワード制御となり，フィードバック制御を全く含んでいないので，雑音やモデル化誤差などに敏感となる．雑音やモデル化誤差に耐性のある制御系を構築するためには，フィードバック制御系を用いる必要がある

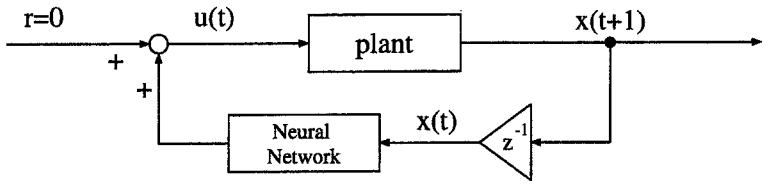


図 2.1: Block diagram of a feedback control system by use of a neural network

が，最適なフィードバック解を求めることは線形システム，二次形式評価関数かつ拘束条件のない場合以外は不可能である．そこで，最適フィードバック解を求めることを目的として，問題を定式化する．まず，ここでは取り扱う問題をシステムの状態を原点に遷移させるレギュレータ問題であるとする．

[問題]

制御対象の状態方程式が

$$\boldsymbol{x}(t+1) \;=\; \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2.1}$$

で表されるとする．ここで，$\boldsymbol{x}(t)$ は n 次元状態変数ベクトル，$\boldsymbol{u}(t)$ は r次元入力変数ベクトル，及び$\boldsymbol{f}$は n 次元ベクトル値関数である．また，システムの原点は不動点であるとする．

$$\boldsymbol{f}(\boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{0} \tag{2.2}$$

このようなシステムに対して，次の評価関数 Jを最小にするような最適フィードバック制御入力$\boldsymbol{u}(t)$ を求める．

$$J \;=\; \sum_{t=1}^{T} J_0(\boldsymbol{x}(t), \boldsymbol{u}(t-1)) \tag{2.3}$$

$$\boldsymbol{u}(t) = \boldsymbol{g}(\boldsymbol{x}(t)) \tag{2.4}$$

ここで，J_0はスカラー値非負関数である．もし，(2.1) における$\boldsymbol{f}$は未知であるときは，まずシステム (2.1) の同定をニューラルネットワークを用いて行う．以下では，制御対象 (2.1) は既知，またはその特性を十分に学習したニューラルネットワークが存在すると仮定する．

2.2.2 フィードバック制御器としてのニューラルネットワークの性質

図 2.1にニューラルネットワーク用いた状態フィードバック制御系を示す．ここで，最適フィードバック制御 (2.4) をニューラルネットワークの学習により実現することが目的である．また，状態フィードバック制御系を考えるために，動的な制御系を考えなくてよい．このためにフィードバック制御器として階層構造型ニューラルネットワークを用いることが可能である．また，[1] の結果から，中間層が 1 層である階層構造

型ネットワークで十分である．明らかなように，ニューラルネットワークへの入力はシステムの状態$\boldsymbol{x}(t)$，出力は制御入力$\boldsymbol{u}(t)$である．

ここでは，制御器として用いるニューラルネットワークの性質を考える．制御対象の原点は不動点であり，システムが原点にあるときに制御入力$\boldsymbol{u}$が$\mathbf{0}$でなければ，一般に状態は原点から遷移する．よって，ニューラルネットワークに$\mathbf{0}$が入力されたときの出力が$\mathbf{0}$でないとすると，定常偏差や自励振動など悪影響が生じ，原点を安定に保つことができない．よって，フィードバック制御器として用いるニューラルネットワークは，$\mathbf{0}$-入力のときに$\mathbf{0}$-出力とならねばならない．しかし，階層構造型ニューラルネットワークに$\mathbf{0}$が入力されたときの出力は一般に$\mathbf{0}$ではない．さらに，ニューラルネットワークを評価関数(2.3)を学習の評価関数として学習を行なうとき，$\mathbf{0}$-入力のときに$\mathbf{0}$-出力という条件は学習の評価に陽ではなく陰に含まれている条件となる．よって，制御系の学習において$\mathbf{0}$-入力において$\mathbf{0}$-出力となることを学習の拘束条件として陽に加えないときには，このように陰的に加わっている条件の学習が成功することに期待せねばならない．しかし，階層構造型ニューラルネットワークの計算を数値的に行う際には，数値誤差の影響もあるために$\mathbf{0}$-入力のときに$\mathbf{0}$-出力はほとんど満足されない．よって，一般の階層構造型ニューラルネットワークを用いると学習がうまくすすまなかったり，あるいは定常偏差や原点付近の小さな自励振動を引き起こす．よって，ニューラルネットワークの入出力関係式を変更することにより問題を解決することを考えねばならない．階層構造型ニューラルネットワークの入出力関係式は(2.5)で表される．

$$\begin{aligned} u_i(t) &= F_O(\sum_{j=1}^{N} \beta_{ij} H_j(t)) \\ &= F_O(\sum_{j=1}^{N} \beta_{ij} F_H(\sum_{k=1}^{n} \alpha_{jk} F_I(x_k(t)) + \theta_i)) \end{aligned} \tag{2.5}$$

入出力関係式(2.5)において$\boldsymbol{x}(t) = 0$を代入し，ネットワークに$\mathbf{0}$が入力されたときの出力を求めると(2.6)を得る．

$$u_i(t) \;=\; F_O(\sum_{j=1}^{N} \beta_{ij} F_H(\sum_{k=1}^{n} \alpha_{jk} F_I(0) + \theta_j)) \tag{2.6}$$

学習によって各活性化関数は変化しないが，結合度や中間ユニットのしきい値は学習により変化する．そこでまず，結合度やしきい値の変化によりネットワークの$\mathbf{0}$-入力の際の出力は変化しないための条件を考える．まず，結合度αの影響を取り除くには入力ユニットの活性化関数として原点を通るものを選べばよいことは(2.6)より自明である．次に，中間ユニットのしきい値θの影響を考える．中間ユニットの活性において，入力ユニットや出力ユニットと同様にしきい値を用いないとすると，ニューラルネットワークによる関数の近似の精度を著しく低下する．よって，中間ユニットの活性においてしきい値θは必要である．中間ユニットのしきい値θの影響を無くすには，活性化(1.3)におけるしきい値の取り扱いの変更が必要であり，

このためには (2.6) より (2.7) と変更するとよいことが分かる．

$$H_i(t) \;=\; F_H(net_i^H(t)) - F_H(\theta_i) \tag{2.7}$$

よって，入力ユニットの活性化関数として原点を通るものを選び，中間ユニットを (2.7) を用いて活性化するより，**0**-入力のときの出力ユニットにおける入力は必ず 0 となり，学習による変化の影響は無くなる．このときのネットワークの出力は $F_O(0)$ であるので，出力ユニットの活性化関数 F_O として原点を通るものを選ぶと，**0**-入力のときの出力は必ず **0** となる．以上をまとめると，以下の通りである．

1. 入力層の活性化関数 F_I と出力層の活性化関数 F_O として原点を通るものを選ぶ．

2. 中間層に含まれるユニットの活性化の式として (2.7) を採用する．

このような階層構造型ニューラルネットワークを用いると，いかなる結合度であっても **0**-入力のときの出力は必ず **0** となり，ニューラルネットワークを用いたフィードバック制御系の原点は不動点となる．

2.2.3 学習アルゴリズム

本節では，ニューラルネットワークを用いてフィードバック制御器の学習を行う学習アルゴリズムに関して考察する．(2.4) の形の最適フィードバック解が解析的に得られるのは制御対象が線形であり，二次形式評価関数かつ拘束条件がない場合に限られる．しかし，一般の場合には最適フィードバック解は解析的に得ることはもちろん，数値計算を用いて入力時系列として得ることも困難である．よって，教師信号を与えることができないために，ニューラルネットワークの学習アルゴリズムとして誤差逆伝播法 [3] を用いることができない．数値計算により得られた最適制御入力時系列を用いて学習を行なうことにより，最適フィードバック制御系をニューラルネットワークにより構築する研究 [24] もあるが，これは間接的な学習により最適フィードバック制御器を構築を行うものであるが，このためには非常に多くの教師データを作成することが必要となり，非常に多くの初期条件のもとで最適制御問題を数値的に解かねばならない．また，数値計算により得られる最適入力は一般には時間の関数となる．このために時間を陽に入力されないニューラルネットワークを用いて学習することができるかが不明確となる．しかし，この方法の長所として学習アルゴリズムとして誤差逆伝播法が利用できる点が挙げられるが，非常に多くの繰り返し計算が必要が必要となっている [24]．このような短所の原因は間接的な学習にあると考えられる．よって，ニューラルネットワークの直接的な学習により (2.4) のような最適なフィードバック制御器を構築方法が必要である．しかし，直接的な学習では教師信号を用いることはできず，直接的に評価関数の最適化を学習により行う必要がある．このためには学習アルゴリズムとして誤差逆伝播法を用いることができず，新たな学習アルゴリズムを導出する必要がある．

2.3 勾配法に基づく学習アルゴリズム

2.3.1 フィードバック制御系の学習法

2.2.3節で述べたように最適フィードバック制御系の構築に対してニューラルネットワークの学習を適用する場合，望ましい応答，すなわち教師信号が与えられないために，誤差逆伝播法を用いることができない．1.2.1節で示したように，誤差逆伝播法は評価関数の勾配を計算するためのものであったが，勾配の計算に立ち戻って学習アルゴリズムの導出を行なう [12]．勾配を計算することを可能とするために，本節ではニューラルネットワークの活性化関数やシステムの状態方程式である (2.1)，評価関数である (2.3) は議論の範囲内は各変数で微分可能であるものとする．フィードバック制御器として用いるニューラルネットワークにおける各結合度による評価関数 Jの微分，すなわち，Jの結合度空間における勾配ベクトルが求めることができれば，最急降下法などの最適化アルゴリズムにより評価関数の最適化する学習アルゴリズムを構築することが可能となり，この学習アルゴリズムを用いて学習を行なったニューラルネットワークにより，評価関数 Jを最適とするフィードバック制御系を実現することができる．

2.3.2 状態方程式が未知の場合

まず，システム状態方程式が未知であるとする．このような場合には，あらかじめ [4,12,14,16] において示されている方法により，ニューラルネットワークにより制御対象のモデルをあらかじめ実現することが必要となる．フィードバック制御系の学習には，図 2.1に示した制御系において，制御対象をモデル化に用いたニューラルネットワークと置き換えたものを用いる．このとき，フィードバック制御器として用いるネットワークと制御対象のモデルとして得られたネットワークが結合することにより，全体としてひとつの複合化されたニューラルネットワークとなる．図 2.1の制御系はひとつの大きなニューラルネットワークとなる．図 2.2に複合ニューラルネットワークを示す．よって，最適フィードバック制御系の学習は図 2.2の複合ニューラルネットワークにおける学習となる．以下の説明ではネットワークの各結合度は図 2.2と同じ記号とする．

複合ニューラルネットワークは二つの階層構造型ニューラルネットワークを接続したものであるので，各ユニットは階層を形成する．図 2.2によると，中間層が 3 層である階層構造型ニューラルネットワークのように思われるが，一般の階層構造型ニューラルネットワークと異なり，制御対象モデルとしてのニューラルネットワークの出力が時間遅れ要素を経てフィードバック制御器のニューラルネットの入力や制御対象モデルのニューラルネットワークの入力の一部としてフィードバックされている．よって，複合ニューラルネットワークは階層構造型ではなく，相互結合型ニューラルネットワークの一種である．しかし，相互結合

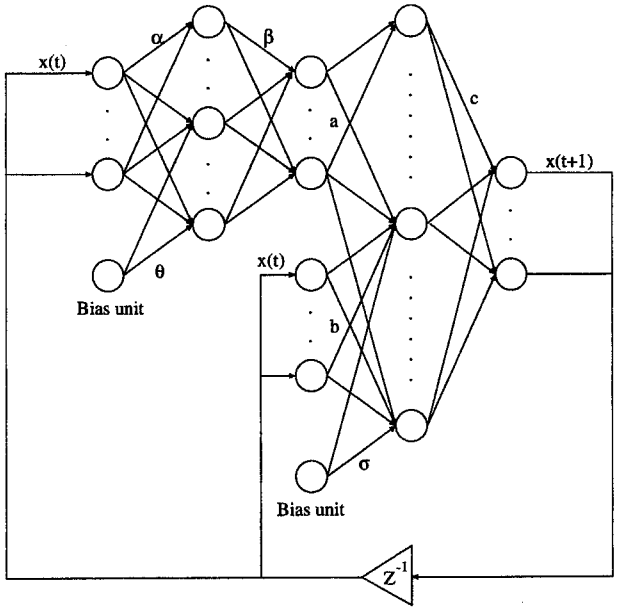


図 2.2: Mixed neural network

型ネットワークの代表例であるホップフィールドモデルやボルツマンマシーンと比較すると，その構造は簡単であり，階層構造型との中間的な構造を持っているといえる．フィードバック制御器および，制御対象のモデルのニューラルネットワークのそれぞれの入出力関係式は (2.8)，(2.9) である．ここでは，フィードバック制御器のニューラルネットワーク，および制御対象のモデルであるニューラルネットワークにおける中間層のユニットの数をそれぞれ N個，M個であるとした．

$$\begin{aligned} u_i(t) &= F_O(\sum_{j=1}^{N} \beta_{ij} H_j(t)) \\ &= F_O(\sum_{j=1}^{N} \beta_{ij}(F_H(\sum_{k=1}^{n} \alpha_{jk}\ F_I(x_k(t)) + \theta_j)\ -\ F_H(\theta_j))) \end{aligned} \tag{2.8}$$

$$\begin{aligned} x_i(t+1) &= F_O(\sum_{j=1}^{M} c_{ij} T_j(t)) \\ &= F_O(\sum_{j=1}^{M} c_{ij}(F_T(\sum_{k=1}^{r} a_{jk} u_k(t)\ +\ \sum_{k=1}^{n} b_{jk} x_k(t)))) \end{aligned} \tag{2.9}$$

(2.9) に (2.8) を代入することにより，図 2.2の複合ニューラルネットワークの入出力関係式が得られる．

フィードバック制御系の学習では，制御対象をモデル化したニューラルネットワークの結合度は不変である．よって，フィードバック制御系として用いるニューラルネットワークに含まれる結合度 wの勾配のみ

を考えればよい．複合ニューラルネットワークにおける Jの勾配関数$\partial J/\partial w$は

$$\frac{\partial J}{\partial w} = \sum_{t=1}^{T}\left[\sum_{j=1}^{n}\frac{\partial x_j(t)}{\partial w}\frac{\partial J_0}{\partial x_j(t)} + \sum_{j=1}^{r}\frac{\partial u_j(t-1)}{\partial w}\frac{\partial J_0}{\partial u_j(t-1)}\right] \tag{2.10}$$

と書ける．よって，$\partial J/\partial w$を求めることは，$\partial \boldsymbol{x}(t)/\partial w$, $\partial \boldsymbol{u}(t)/\partial w$ を求めることに帰着される．まず，$\partial \boldsymbol{u}(t)/\partial w$ を計算する．通常の階層構造型ニューラルネットワークと異なり，ネットワークの入力である$\boldsymbol{x}(t)$ も wの関数であることに注意して，計算を実行すると (2.11)，(2.12)，(2.13) を得る．

$$\frac{\partial u_i(t)}{\partial \alpha_{pq}} = F_O'(net_i^O)\left\{\beta_{ip}F_H'(net_p^H)I_q(t) + \sum_{j=1}^{N}\beta_{ij}F_H'(net_j^H)\sum_{k=1}^{n}\alpha_{jk}F_I'(x_k(t))\frac{\partial x_k(t)}{\partial \alpha_{pq}}\right\} \tag{2.11}$$

$$\frac{\partial u_i(t)}{\partial \beta_{pq}} = F_O'(net_i^O)\left\{\delta_{ip}H_q(t) + \sum_{j=1}^{N}\beta_{ij}F_H'(net_j^H)\sum_{k=1}^{n}\alpha_{jk}F_I'(x_k(t))\frac{\partial x_k(t)}{\partial \beta_{pq}}\right\} \tag{2.12}$$

$$\frac{\partial u_i(t)}{\partial \theta_p} = F_O'(net_i^O)\left\{\beta_{ip}\left(F_H'(net_p^H) - F_H'(\theta_p)\right) + \sum_{j=1}^{N}\beta_{ij}F_H'(net_j^H)\sum_{k=1}^{n}\alpha_{jk}F_I'(x_k(t))\frac{\partial x_k(t)}{\partial \theta_p}\right\} \tag{2.13}$$

ここで F_I', F_H'及び F_O'はそれぞれの活性化関数の各変数における微分を表す．ここで (2.11)，(2.12)，(2.13) の右辺第一項に着目すると，時刻 t におけるネットワークの結合度の変化によるものであり，1.2.1節において導いた誤差逆伝播法と同じであることは容易に分かる．これに対して，第二項は時刻 t 以前の結合度の変化による状態変数の変化による項であり，従来の誤差逆伝播法には存在しない．この項は複合ニューラルネットワークが相互結合型であるために現れた項である．以上により，(2.11)，(2.12)，(2.13) から，$\partial \boldsymbol{u}(t)/\partial w$ を求めることは，$\partial \boldsymbol{x}(t)/\partial w$を求めることに帰着された．

次に，$\partial \boldsymbol{x}(t+1)/\partial w$を求める．制御対象のモデルであるニューラルネットワークの入力である$\boldsymbol{x}$と$\boldsymbol{u}$はやはり wの関数となる．このことを考慮して計算を実行すると，(2.14)，(2.15)，(2.16) を得る．

$$\frac{\partial x_i(t+1)}{\partial \alpha_{pq}} = \sum_{j=1}^{M}c_{ij}F_T'(net_j^T)\left\{\sum_{k=1}^{r}a_{jk}\frac{\partial u_k(t)}{\partial \alpha_{pq}} + \sum_{k=1}^{n}b_{jk}\frac{\partial x_k(t)}{\partial \alpha_{pq}}\right\} \tag{2.14}$$

$$\frac{\partial x_i(t+1)}{\partial \beta_{pq}} = \sum_{j=1}^{M}c_{ij}F_T'(net_j^T)\left\{\sum_{k=1}^{r}a_{jk}\frac{\partial u_k(t)}{\partial \beta_{pq}} + \sum_{k=1}^{n}b_{jk}\frac{\partial x_k(t)}{\partial \beta_{pq}}\right\} \tag{2.15}$$

$$\frac{\partial x_i(t+1)}{\partial \theta_p} = \sum_{j=1}^{M}c_{ij}F_T'(net_j^T)\left\{\sum_{k=1}^{r}a_{jk}\frac{\partial u_k(t)}{\partial \theta_p} + \sum_{k=1}^{n}b_{jk}\frac{\partial x_k(t)}{\partial \theta_p}\right\} \tag{2.16}$$

以上から明らかなように，$\partial \boldsymbol{x}(t+1)/\partial w$は$\partial \boldsymbol{x}(t)/\partial w$，および$\partial \boldsymbol{x}(t)/\partial w$の関数となる．

ここで得られた (2.11)～(2.16) により，Jの勾配を求めるために必要となる式は，$\partial \boldsymbol{x}(t)/\partial w$の漸化式で表されることが分かる．よって，初期値にあたる$\partial \boldsymbol{x}(0)/\partial w$が与えられれば，$\partial \boldsymbol{x}(t)/\partial w$は各時刻において計算することが可能となり，その結果として勾配 (2.10) を計算することができる．ここで，図 2.2の複合ニューラルネットワークの初期値は制御対象のモデルであるニューラルネットワークの出力層の活性値$\boldsymbol{x}(0)$

である．これは$\boldsymbol{x}(0)$が制御対象の初期値であることから明らかである．また，$\boldsymbol{x}(0)$は制御器が変更することができる量ではないので，結合度wにより初期値$\boldsymbol{x}(0)$は変化することはない．すなわち，

$$\frac{\partial x_i(0)}{\partial \alpha_{pq}} = \frac{\partial x_i(0)}{\partial \beta_{pq}} = \frac{\partial x_i(0)}{\partial \theta_p} = 0 \quad \forall i, \forall p, \forall q \tag{2.17}$$

が成立する．以上により，勾配(2.10)を求めることは，各時刻において$\partial \boldsymbol{x}(t+1)/\partial w$および$\partial \boldsymbol{u}(t)/\partial w$を逐次計算することにより可能であることが示された．このようにして求められた勾配$\partial J/\partial w$から，最急降下法や共役勾配法などの最適化アルゴリズムを用いることにより，最適フィードバック制御系を学習することができる．最適フィードバック制御系の学習が終了がした後，実際の制御対象に学習済のニューラルネットワークを図2.1のように接続することにより，制御対象に対して最適フィードバック制御が施される．

2.3.3 誤差逆伝播法との対応

本節では，誤差逆伝播法と2.3.2節で得られた学習アルゴリズムとの対応を考える．最適化する評価関数として(2.18)を考える．

$$J = \frac{1}{2}\sum_{t=1}^{T}\left\{\sum_{i=0}^{n} x_i^2(t) + \sum_{i=0}^{r} u_i^2(t-1)\right\} \tag{2.18}$$

この評価関数は一般に二次形式といわれるものであるが，$\boldsymbol{x}(t)$と$\boldsymbol{u}(t)$にそれぞれと同じ次元の零ベクトルを教師信号として与えた場合の誤差二乗関数とみなすこともできる．また，Rumelhartの方法[3]に従うと，図2.2に示した複合ニューラルネットワークを階層構造型ネットワークに展開することができる．このように展開することにより得られる階層構造型ネットワークは非常に多くの中間層を持ち，異なる層間結合であっても同じ結合度が用いられているだけではなく，教師信号は出力層だけでなく中間層にも与えられている．誤差逆伝播法が導出された階層構造型ニューラルネットワークでは，同じ結合度を異なる層間で用いることはないうえに，教師信号は出力層にのみに与えられる．よって，2.3.2節で示した学習アルゴリズムは，出力層だけでなく複数の中間層にも教師信号が与えられ，異なる層間であっても同じ結合度を用いることを許す階層構造型ニューラルネットワークにおける誤差逆伝播法の拡張と見ることもできる．

2.3.4 状態方程式が既知である場合

制御対象の状態方程式が既知の場合，あるいは数学モデルが与えられている場合には制御対象をニューラルネットワークにより同定する必要はなく，制御対象あるいはモデルの状態方程式を学習に用いる．2.3.2節で示した勾配計算法において，同定に用いたニューラルネットワークの部分を状態方程式に変更することにより，(2.14)(2.15)(2.16)は次のように変更される．

$$\frac{\partial x_i(t+1)}{\partial \alpha_{pq}} = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j(t)} \frac{\partial x_j(t)}{\partial \alpha_{pq}} + \sum_{j=1}^{r} \frac{\partial f_i}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial \alpha_{pq}} \tag{2.19}$$

$$\frac{\partial x_i(t+1)}{\partial \beta_{pq}} = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j(t)} \frac{\partial x_j(t)}{\partial \beta_{pq}} + \sum_{j=1}^{r} \frac{\partial f_i}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial \beta_{pq}} \tag{2.20}$$

$$\frac{\partial x_i(t+1)}{\partial \theta_p} = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j(t)} \frac{\partial x_j(t)}{\partial \theta_p} + \sum_{j=1}^{r} \frac{\partial f_i}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial \theta_p} \tag{2.21}$$

$\partial \boldsymbol{u}(t)/\partial w$に関しては2.3.2節と同じである．また，同様に初期値$\partial \boldsymbol{x}(0)/\partial w$は，$\boldsymbol{x}(0)$ は明らかに $\mathbf{0}$ である．以上により，2.3.2節と全く同様にして勾配を計算できる．ここで，(2.19)，(2.20)，(2.21) を見ると状態方程式の微分係数，すなわちヤコビアンを求めていることが分かる．よって，(2.14)，(2.15)，(2.16) では制御対象のモデルのニューラルネットワークのヤコビアンを求めて利用していたことが分かる．このように，評価関数の勾配を求めるためには制御対象のヤコビアンを計算することが必須である．

2.3.2節で示した学習では，制御対象をモデル化したニューラルネットワークを用いているが，モデルと制御対象との間にはある程度の誤差が存在する．このようなモデル化誤差のために，学習の際の結果と実際の制御対象を制御を行なったときには評価関数や制御結果に差が生じる．このとき制御性能は劣化することが多いために，モデル化誤差を小さくすることが性能の高い制御系の学習に不可欠である．しかし，制御対象の正確なモデルを得ることは難しく，モデル化誤差を完全に取り除くことは難しい．このように，制御対象にはモデル化誤差に代表される不確さが存在していることが一般的であるが，本節で述べた制御系の学習方法では不確さを考慮していないため，制御系のロバスト性が問題となる可能性がある．

2.3.5 計算例

本節では，勾配法に基づく学習アルゴリズムを用いてさまざまな制御対象に対してニューラルネットワークを用いて最適フィードバック制御系の構築を行なった例を示す．

制御対象 G1

制御対象の状態方程式を

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.3 \\ 0.2 & 0.7 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0.0 \\ 0.2 \end{bmatrix} u(t) \tag{2.22}$$

とする．この制御対象は線形かつ可制御なものである．まず，状態方程式 (2.22) を未知として，フィードバック制御系の学習には制御対象の特性を十分に学習した三層構造ニューラルネットワークを用いた．評価関数，および制御対象の初期状態をそれぞれ (2.23)，(2.24) とした．

$$J = \sum_{t=1}^{50} \left[x_1^2(t) + x_2^2(t) + u^2(t-1) \right] \tag{2.23}$$

$$x_1(0) = 1.0 \quad , \quad x_2(0) = 0.0 \tag{2.24}$$

フィードバック制御系の学習には三層構造ニューラルネットワークを用い，出力層と入力層の活性化関数はともに恒等関数を，中間層の活性化関数には双曲正接関数 tanh を用い，中間ユニット数は 50 個とした．図 2.3に学習が完了したニューラルネットワークによる制御入力，図 2.4にそのときの状態変数の推移を示す．制御対象 (2.22) が既知であるとすると，評価関数 (2.23) を最小にするフィードバック制御系を解析的に得ることが可能である．比較のために，図 2.3，2.4には解析的に解いた最適フィードバック制御による結果も示した．ただし，ここでは最適フィードバック制御器として定常最適レギュレータを用いた．図 2.3，2.4によると，最適レギュレータとほぼ同じ状態フィードバック制御系がニューラルネットワークの学習により得られていることが分かる．本節の手法を用いた場合には，評価関数は 6.776×10^{-1}であったが，最適レギュレータでの評価関数は 6.769×10^{-1}であり，わずかにニューラルネットワークのほうが劣る結果となった．このような差が生じた原因を確認するために，状態方程式 (2.22) が既知として同じ構造のニューラルネットワークを用いて学習を行なった．図 2.5に得られた制御入力を示し，図 2.6にそのときの状態変数の推移を示す．ニューラルネットワークによる制御での評価関数は 6.769×10^{-1}となり，最適レギュレータによるものと一致した．これにより，図 2.3，図 2.4にに存在した差は，ニューラルネットワークによる同定誤差が原因であったことが確認できた．このように，学習に用いた制御対象のモデルに誤差が存在するとき，実際の制御結果は最適とはならない．

制御対象 G2

制御対象の状態方程式を

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} 1.0 & 1.0-e^{-h} \\ 0.0 & e^{-h} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} h-1.0+e^{-h} \\ 1.0-e^{-h} \end{bmatrix} u(t) \tag{2.25}$$

とする．これは，連続系における線形システム (2.26) をサンプリング幅 h で離散化したものである．

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0.0 & 1.0 \\ 0.0 & -1.0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0.0 \\ 1.0 \end{bmatrix} u(t) \tag{2.26}$$

ただし，ここでは h は 0.01 であるとし，サンプリング間の入力は 0 次ホールドされているものとしている．この状態方程式 (2.25) が既知であるとして，ニューラルネットワークの学習を行なった．評価関数，および制御対象の初期状態はそれぞれ (2.27)，(2.28) とした．

$$J = \sum_{t=1}^{800} \left[x_1^2(t) + x_2^2(t) + u^2(t-1) \right] \tag{2.27}$$

$$x_1(0) = 4.5 \ , \ x_2(0) = 0.0 \tag{2.28}$$

活性化関数や中間ユニット数は，2.3.5節と同じであるとした．この制御対象も線形可到達な系であるので解析的に最適なレギュレータが構成できるために，ニューラルネットワークの学習により構築された制御

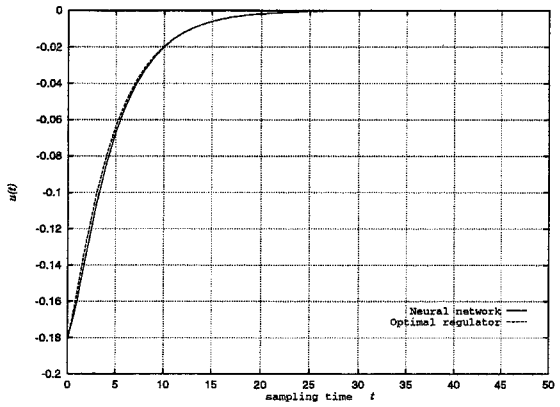


図 2.3: Optimal feedback control(Plant G1)

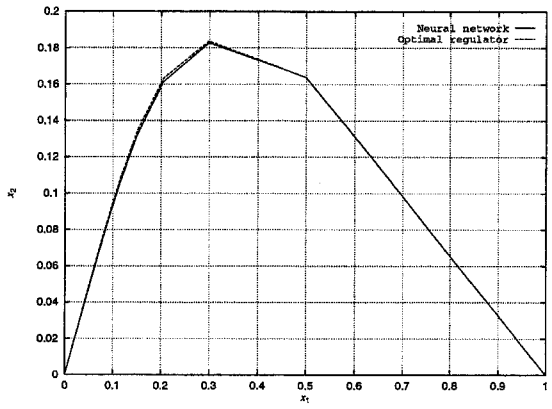


図 2.4: Transtion of the state(Plant G1)

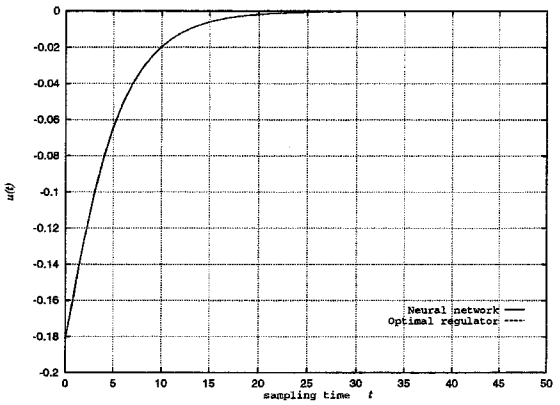


図 2.5: Optimal feedback control(Plant G1)

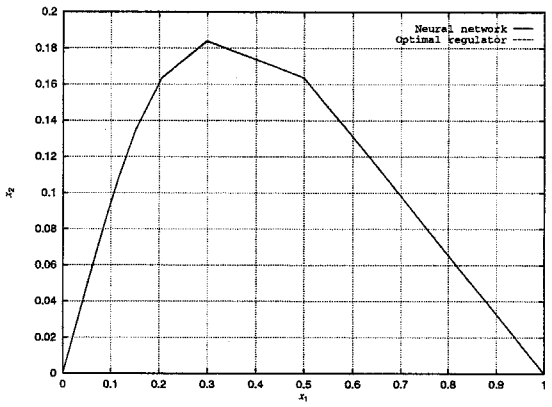


図 2.6: Transtion of the state(Plant G1)

系との比較に用いる．ただし，本節でも解析解としては定常最適レギュレータを用いた．図2.7に得られた制御入力，図2.8にそのときの状態変数の推移を示す．これらの図より，ニューラルネットワークは学習により最適レギュレータを実現していることが分かる．評価関数を調べても最適レギュレータと学習を行なったニューラルネットワークには差はなく，共に 4.040×10^3であった．また，ここで学習済みのニューラルネットワークの汎化性を調べるために，制御対象の初期状態を (2.29) で表される領域内で変更したときの評価関数 (2.27) の変化を調べた．

$$3.0 \ \leq \ x_1(0) \ \leq \ 6.0 \ , \ -1.5 \ \leq \ x_2(0) \ \leq \ 1.5 \tag{2.29}$$

図2.9に領域 (2.29) におけるニューラルネットワークによる評価関数と最適レギュレータによる評価関数の比を示す．図2.9から分かるように，領域 (2.29) 内では最適レギュレータからの誤差が1%未満になっており，学習後のニューラルネットワークが十分な汎化能力を有していることが示された．

さらに，評価関数を (2.30) のように変更し，同じ構造のニューラルネットワークを用いて学習を行なった．

$$J \ = \ \sum_{t=1}^{800} \left[x_1^2(t) \ + \ x_2^4(t) \ + \ u^2(t-1) \right] \tag{2.30}$$

評価関数 (2.30) と (2.27) の違いは，$x_2(t)$ の次数だけである．特に，(2.30) では $x_2(t)$ の次数が4次であるので，$x_2(t)$ の絶対値が1を越えると急速に評価関数は大きくなる．このために，$|x_2(t)| \leq 1$ という拘束条件を緩やかに加える効果 [25] がある．しかし，評価関数 (2.30) は二次形式でないため，たとえ定常解であっても解析的に解くことは不可能となる．しかし，ニューラルネットワークの学習を行うことにより，このような評価関数を最小にする最適フィードバック制御系を構築することが可能となる．図2.10に得られた制御入力，図2.11にそのときの状態変数の推移を示す．Rekasius [25] によると，評価関数が (2.30) のときの最適な制御入力の近似解は次のように得られている．

$$u(t) \ = \ -x_1 + 0.732 x_2 + x_2 \left(0.067 x_1^2 - 0.115 x_1 x_2 + 0.267 x_2^2 \right) \tag{2.31}$$

ただし，これは制御対象を連続時間システムとして導いたものであり，離散時間系において導いたものではない．(2.31) による制御結果を比較のためにそれぞれの図に示した．ここで $x_2(t)$ に注目すると，図2.8に比べ図2.11のほうが $x_2(t)$ の絶対値が小さくなっていることは明らかであり，緩やかな拘束がうまくはたらいていることが分かる．評価関数の値はニューラルネットワークによる解は 4.224×10^3であり，(2.31) の近似最適フィードバック制御では 4.266×10^3であった．評価関数の大きさから判断して，ニューラルネットワークの方が優れた性能であったことが分かる．一般に，近似解 (2.31) を得ることは大変困難であるのに対して，ニューラルネットワークの学習によると最適フィードバック制御系を容易に得ることができる．また，学習済みのニューラルネットワークの汎化性を調べるために，制御対象の初期状態を (2.29) のような領

域内で変化させたときの評価関数 (2.30) の変化を調べた．図 2.12に領域 (2.29) におけるニューラルネットワークによる評価関数と近似解による評価関数の比を示す．この図より領域 (2.29) において学習後のニューラルネットワークは近似解よりも高い性能を持つことが分かり，ニューラルネットワークの学習による制御系構築の有効性が示された．

制御対象 G3

制御対象を

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ 0.2(1.0 - x_1^2(t))x_2(t) - x_1(t) + u(t) \end{bmatrix} \tag{2.32}$$

をサンプリング幅 0.01 で Euler 差分化したものとする．これは Van del Pole の方程式と呼ばれる原点不安定だが安定なリミットサイクルを持つ系である．この状態方程式を既知であるとして，ニューラルネットワークの学習により制御系の構築を行なった．この制御対象は非線形なものであるので，従来の方法で解析解を求めることはできない．さらに，入力に拘束条件が存在する場合には制御系設計問題はさらに困難となるが，ニューラルネットを用いることで最適な制御系を学習により構築することができる．評価関数，および制御対象の初期状態をそれぞれ (2.33)，(2.34) とした．

$$J = \sum_{t=1}^{T} \left[x_1^2(t) + x_2^2(t) + u^2(t-1) \right] \tag{2.33}$$

$$x_1(0) = 0.5 \ , \ x_2(0) = 0.0 \tag{2.34}$$

ただし，制御入力 u には次のような拘束条件があるとする．

$$|u(t)| \leq U \tag{2.35}$$

ここで Uは正のある与えられた定数である．ここまでに示した例と同じように出力層の活性化関数に恒等関数を用いると，学習の結果として拘束条件 (2.35) を破る制御系が得られることがある．そこで出力層の活性化関数をシグモイド関数を用いることにより，ニューラルネットワークが近似できる関数空間を拘束条件 (2.35) を満足する空間に制限する．ここでは，出力層の活性化関数として (2.36) を用いた．

$$F_O(x) = \begin{cases} \frac{3}{2U^2}\left(U^2 x - \frac{x^3}{3}\right) & |x| \leq U \\ \\ U \cdot sgn(x) & |x| > U \end{cases} \tag{2.36}$$

この活性化関数は $x = \pm U$において微分係数が連続となるようにつながっているので，勾配を求めることが可能である．ここでは，Uを 0.4，0.3，0.2，0.1 として学習を行なった．ただし，終端時間である Tは Uが 0.4 及び 0.3 のときには 500 とし，Uが 0.2 及び 0.1 の時には 2000 であるとした．これは入力の拘束条件が

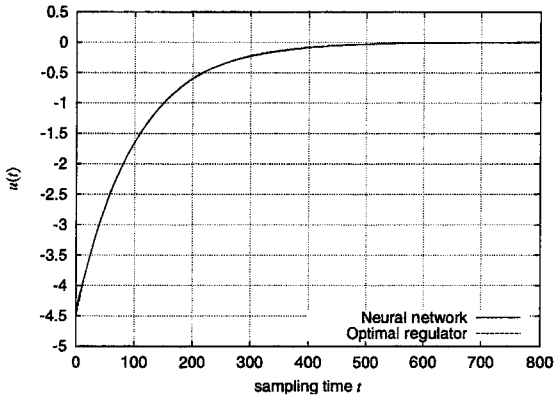


図 2.7: Comparison between a trained neural network and by a optimal regulator(Plant G2)

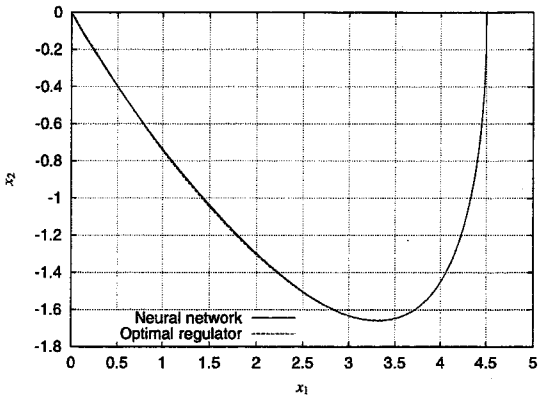


図 2.8: Transtion of the state(Plant G2)

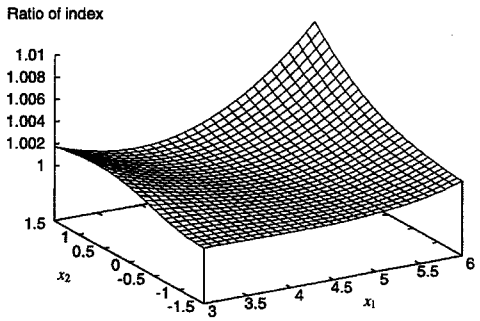


図 2.9: Performance indices from various initial states(Plant G2)

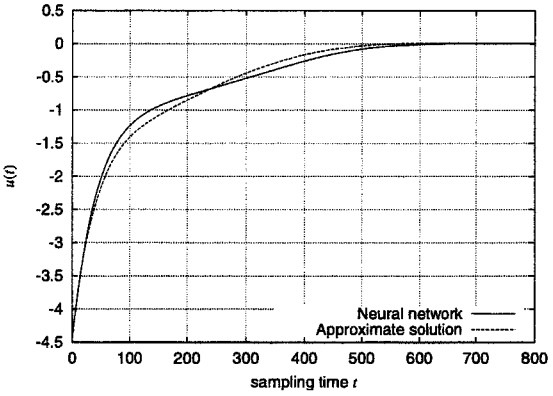


図 2.10: Comparison between a trained neural network and an approximated optimal feedback controller(Plant G2)

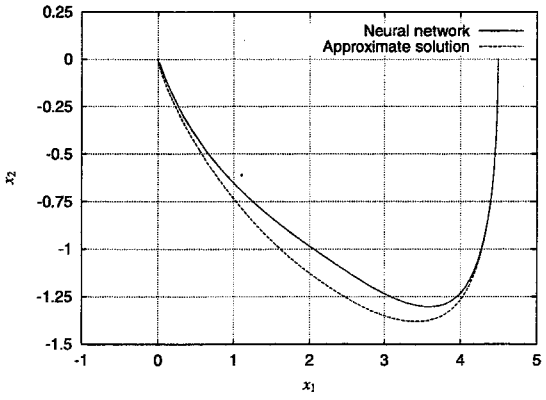


図 2.11: Transtion of the state(Plant G2)

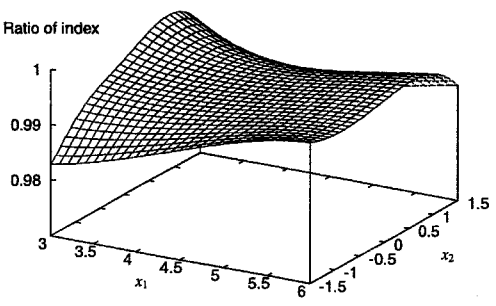


図 2.12: Performance indices from various initial states(Plant G2)

厳しいときには，状態を原点に持っていくために長い時間を要したからである．図 2.13にそれぞれの制御入力を，図 2.14にその際の状態変数の推移を示す．図 2.13によると，$U = 0.4$ のときには入力の拘束条件にかからないが，Uが小さくなるにしたがって拘束条件にかかるようになり，それに伴って状態変数の収束性は悪化していくことが分かる．なお，$U = 0.4$ のときの結果は出力層に恒等関数を用いて入力の拘束条件を付加せずに学習した結果と一致した．さらに，ニューラルネットワークの汎化性を調べるために，入力の拘束がないと仮定し，初期状態 (2.34)，評価関数 (2.33) として出力層に恒等関数を用いて学習をおこなった．学習後のネットワークを用いて，制御対象の初期状態を領域 (2.37) 内で変化させたときの評価関数 (2.33) を調べた．

$$0.0 \leq x_1(0) \leq 1.0\ ,\ -0.5 \leq x_2(0) \leq 0.5 \tag{2.37}$$

その結果を，制御対象の線形近似モデルに基づいて構築した最適レギュレータによる制御結果と比較する．図 2.15に (2.37) におけるニューラルネットワークによる評価関数と近似最適レギュレータによる評価関数の比を示す．$x_1(0) = x_2(0) = 0$ ではニューラルネットワーク，近似最適レギュレータ共に $u(t) = 0$ となり，その結果としていずれの場合も評価関数は 0 になるので，図 2.15では $x_1(0) = x_2(0) = 0$ における評価関数比を 1.0 と設定した．これが図 2.15の原点において評価関数の比が急に大きくなったように見える原因であるが，同時に初期状態が原点に近くても，近似最適レギュレータが最適な制御とは言えないこと意味している．また，図 2.15より学習に用いた初期状態 (2.33) の近くではニューラルネットワークは優れた汎化能力を持っていることが分かるが，制御対象の初期状態が学習に用いたものから遠ざかるにつれて，近似最適レギュレータの方が評価関数が小さくなっており，ニューラルネットワークが最適フィードバック制御器を学習しているとはいえなくなっていることも分かる．この欠点を解消する簡単な方法として，ネットワークの学習において二つの初期状態 (2.34)，(2.38) を用いた．

$$x_1(0) = 0.0\ ,\ x_2(0) = 0.5 \tag{2.38}$$

図 2.16に学習が完了したニューラルネットワークによる (2.37) における評価関数の比を示す．図 2.15と図 2.16を比較すると，二つの初期状態を学習することにより，ニューラルネットワークの汎化能力が向上したことが分かる．このように学習に複数の初期点を用いることにより，解の初期値依存性を軽減され，ニューラルネットワークの汎化能力は高まることが分かる．

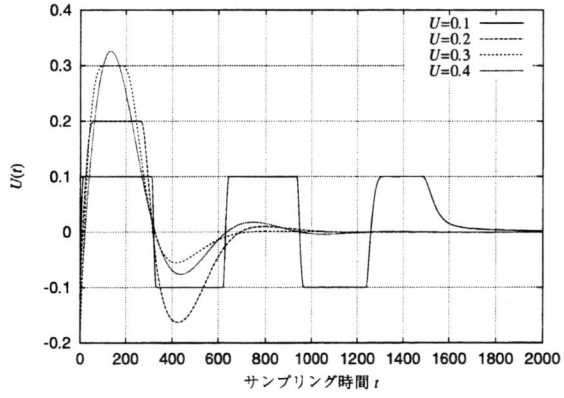


図 2.13: Optimal feedback controls for various U(Plant G3)

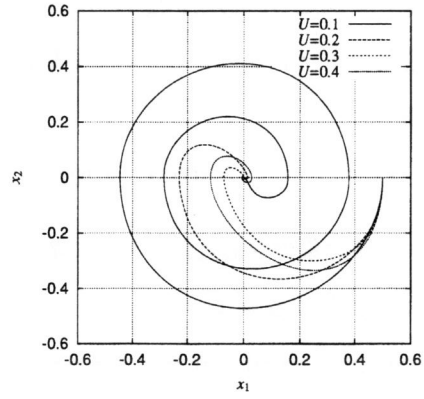


図 2.14: State transitions by each optimal controllers(Plant G3)

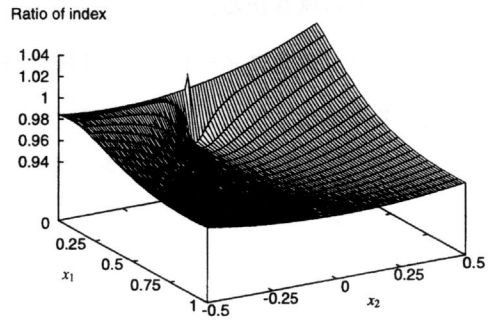


図 2.15: Performance indices from various initial states(Plant G3)

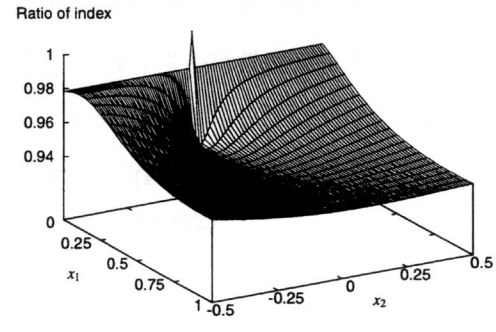


図 2.16: Performance indices from various initial states(Plant G3)

制御対象 G4

制御対象の状態方程式を (2.39) とし，学習にはその数学モデルとして (2.39) をサンプリング幅 0.01 で Euler 差分化したものを用いる．

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_3 + (x_3 - \sin x_3) + x_1 x_4^2 \\ x_4 \\ u \end{bmatrix} \tag{2.39}$$

ここで，実際の制御対象としては (2.39) を 4 次の Runge-Kutta 法で解いたものとする．ただし，入力は 0 次ホールドされるものとする．評価関数と制御対象の初期状態を

$$J = \sum_{t=1}^{1000} \left[x_1^2 + x_2^2 + x_3^2 + x_4^2 + u^2(t-1) \right] \tag{2.40}$$

$$x_1(0) = 2.0 \ , \ x_2(0) = x_3(0) = x_4(0) = 0.0 \tag{2.41}$$

とした．ここでは，中間ユニットの数は 20 個であるとし 2.3.5節と同じ活性化関数を用いた．この制御対象も非線形であるため，最適フィードバック制御問題を解析的に解くことは不可能である．そこで状態方程式を線形近似を行ない，評価関数 (2.40) が最小になるように設計した最適レギュレータを求めて，学習を行なったニューラルネットワークとの比較を行なう．図 2.17に制御入力を，図 2.18，2.19にそのときの状態変数の推移を示す．評価関数は学習が完了後のニューラルネットによる制御結果では 1.490×10^3 近似レギュレータによる制御結果では 1.915×10^3となった．これらの図を見れば分かるように線形近似解には peaking 現象が見られ，この対象の制御を困難なものとしている．peaking 現象は対象の線形近似モデルでも現れる現象であるが，線形モデルであれば最適レギュレータ問題を解いているために，制御系の全ての固有値は左半平面にあり，線形モデルは安定であった．しかし，線形モデルでは無視している発散性ある項 $x_1 x_4^2$が peaking 現象ために大きくなり無視することができなくなり，ついには発散してしまうことが考えられる．しかし，ニューラルネットワークの学習により得られた制御系では，顕著な peaking 現象は抑えられていることが分かる．これは，ニューラルネットワークが学習により，peaking 現象を抑制しなければ評価関数が大きくなってしまうことを学んだからであろう．ニューラルネットワークが peaking 現象を抑制していることを確認するために，初期値を

$$x_1(0) = 2.5 \ , \ x_2(0) = x_3(0) = x_4(0) = 0 \tag{2.42}$$

と変化させたときの学習済みニューラルネットワークと線形近似解による制御結果を比較する．$x_1(t)$ の応答を図 2.20に示す．線形近似解による制御では peakling 現象により，状態は不安定になってしまっているの

に対して，ニューラルネットワークによる制御系によると対象を安定に制御することが可能であった．このような不安定な対象に対しても，ニューラルネットワークの学習により制御を行なう有効性が確認された．

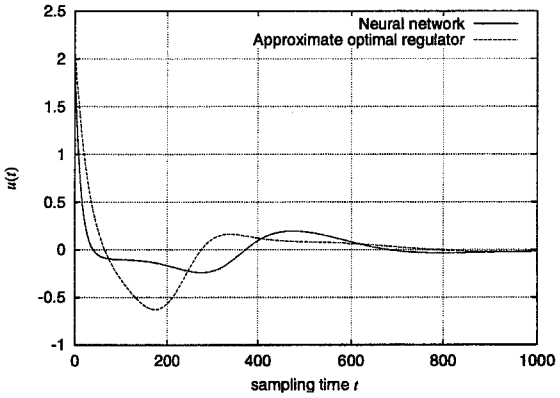


図 2.17: Comparison between a trained neural network and an approximated optimal feedback controller(Plant G4)

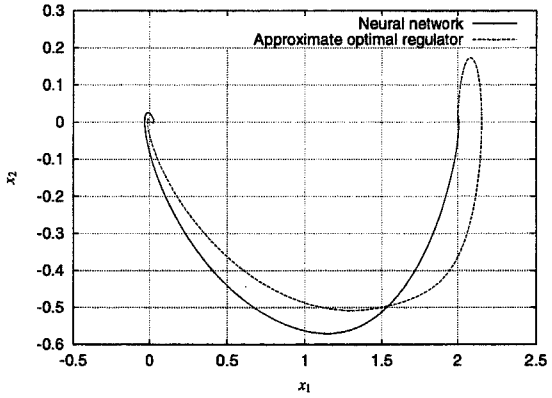


図 2.18: Transtion of the state(Plant G4)

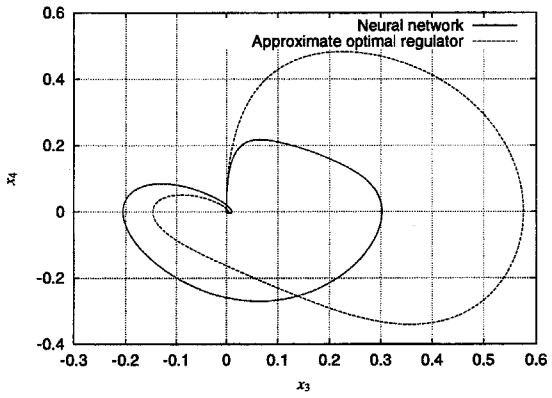


図 2.19: Transtion of the state(Plant G4)

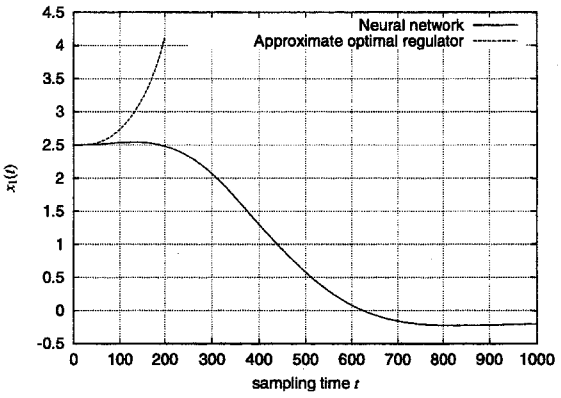


図 2.20: Transtion of the state(Plant G4)

2.4 Powellの共役方向法に基づく学習アルゴリズム

2.4.1 勾配法に基づく学習の欠点

2.3節において勾配法に基づく学習アルゴリズムを導出したが，この方法では評価関数の微分である勾配関数を求める必要があり，このためには評価関数や活性化関数の微分可能性を仮定する必要があった．よって，微分可能でないような評価関数を持つことができないが，ペナルティ法によって状態量拘束を考慮する

場合でも評価関数が微分可能でなくてはならないために，用いることができるペナルティ項は微分可能な滑らかなものに限られるため，拘束を破る解が学習により求められる可能性があった．また，フィードバック制御器として用いるニューラルネットワークは階層構造型であったが，制御対象を含めた制御系全体は相互結合型ネットワークとなり，評価関数の勾配を求めるには微分係数の漸化式を解く必要がある．このために，制御対象を変更する場合や用いるネットワークの構造を変更するには，勾配を求める漸化式を求め直した後に，プログラムの勾配を求める部分はすべて書き直す必要が生じる．以前の計算機では，勾配を求めるために必要な漸化式を解くのために多くの時間が必要であったが，最近のCPUの進歩により計算に必要な時間は大幅に短縮された．しかし，勾配法に基づく学習アルゴリズムでは，問題に応じて人間が勾配を求める式を計算し，学習用のプログラムを変更する必要があるので，学習を実行する準備の手間が大きい．また，書き直したプログラムのデバッグなどに要する時間も必要である．さらに，制御対象の特性を完全に知らねば，学習のアルゴリズムを記述できないことも欠点といえる．このために，制御対象の特性パラメータなどが全て公開されていないと，学習により制御系を構築することができない．

2.4.2 Powellの共役方向法

非線形計画法において最小化する関数の勾配が得られない，あるいは得ることが非常に難しいような場合における最小化の手法で最も有力な方法の一つにPowellの共役方向法がある [26,22]．この方法は勾配，すなわち最急降下方向を求めることなしに最小化したい関数の共役な方向を生成して関数の最小化を行なうものである．n次元ベクトル$\boldsymbol{x}$とそのスカラー値関数$H(\boldsymbol{x})$を最小にする$\boldsymbol{x}$を求める問題を考えると，Powellの共役方向法によるアルゴリズムは以下のように表すことができる．

Powellの共役方向法アルゴリズム

1. n個の独立な方向ベクトル$\boldsymbol{s}_0, \boldsymbol{s}_1, \cdots, \boldsymbol{s}_{n-1}$と出発点$\boldsymbol{x}_0$を与える．

2. $i = 0, 1, \cdots, n-1$について$H(\boldsymbol{x}_i + \lambda \boldsymbol{s}_i)$が最小になるような$\lambda = \lambda_i$を求め，

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \lambda_i \boldsymbol{s}_i \quad i = 0, 1, \cdots, n-1$$

   とおく．

3. $i = 0, 1, \cdots, n-1$について$H(\boldsymbol{x}_i) - H(\boldsymbol{x}_{i+1})$が最大になるような整数$i = m$を求め，$\Delta = H(\boldsymbol{x}_m) - H(\boldsymbol{x}_{m+1})$とおく．

4. $H_3 = H(2\boldsymbol{x}_n - \boldsymbol{x}_0)$を計算し$H_1 = H(\boldsymbol{x}_0), H_2 = H(\boldsymbol{x}_n)$とおく．

5.

$$H_3 \geq H_1 \tag{2.43}$$

あるいは

$$2(H_1 - 2H_2 + H_3)(H_1 - H_2 - \Delta)^2 \geq \Delta(H_1 - H_3)^2 \tag{2.44}$$

が成立する時にはx_nを次の繰り返しの初期点x_0として用い，方向ベクトル$s_0, s_1, \cdots, s_{n-1}$は変更せずに2. に戻る．

6. (2.43)，(2.44) が共に成立しない場合には$\boldsymbol{s}_n = \boldsymbol{x}_n - \boldsymbol{x}_0$ として $H(\boldsymbol{x}_n + \lambda \boldsymbol{s}_n)$ が最小になるような $\lambda = \lambda_n$ を求め

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \lambda_n \boldsymbol{s}_n$$

を次の繰り返しの初期点とする．また，方向ベクトルとして$\boldsymbol{s}_0, \boldsymbol{s}_1, \cdots, \boldsymbol{s}_{m-1}, \boldsymbol{s}_{m+1}, \cdots, \boldsymbol{s}_{n-1}, \boldsymbol{s}_n$ を採用する．

7. εをある微小な正数として繰り返しのときに

$$||H(\boldsymbol{x}_n) - H(\boldsymbol{x}_0)|| \leq \varepsilon$$

が満たされるならば計算を打ち切る．

2.4.3 ニューラルネットワークの学習への適用

ニューラルネットワークの学習への適用を考えると，Powell の共役方向法では一次元探索を行なう必要があるため，オフラインの学習アルゴリズムとして用いることは可能であるが，オンライン学習アルゴリズムとしては適用することができない．しかし，本研究ではニューラルネットワークのオフライン学習により制御系を構築する方法を論じるために，この方法を学習アルゴリズムとして用いることができる．しかし，ネットワークの入出力関係式に着目すると，シグモイド型活性則を持つニューラルネットワークの学習に用いる際には，注意が必要であることが分かる．シグモイド型の活性則によるネットワークの入出力関係式を書き下すと，(2.45) となる．

$$O_i = \beta_{ij} F_H(\alpha_{jk} I_k + \theta_j) \tag{2.45}$$

ここで，学習によりO_iが大きくなると望ましい結果となると仮定すると，通常の一次元探索のように無制限に探索方向に結合度を増加させることが可能であるならば，α_{jk}の方向を探索するときに，I_kと同符号の無限大まで探索することになる．このようなことは，入力層と中間層との間の結合度の出力への影響は，中間層の活性化関数であるシグモイド関数により有界であるために生じる．よって，一次元探索により結合度を

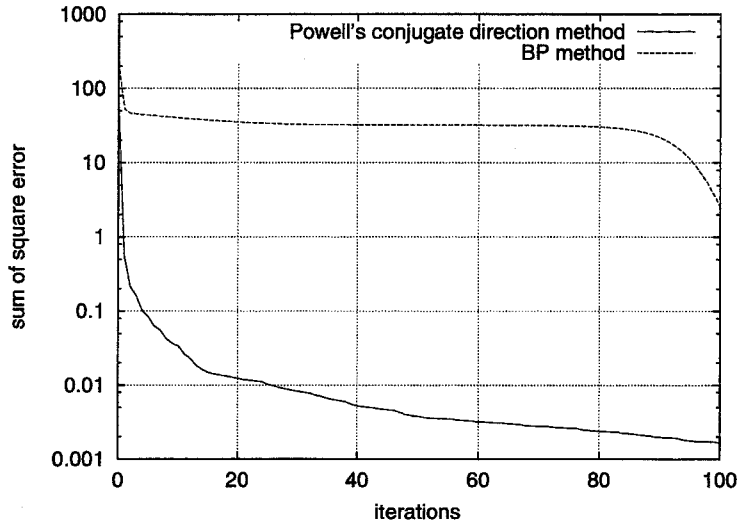


図 2.21: Comparison between back propagation algorithm and Powell's conjugate direction algorithm

増加することができる回数を制限する必要がある．しかし，このような簡単な制限を行なうだけで，Powell の共役方向法はニューラルネットワークのオフライン学習法として用いることが可能となる．Powell の共役方向法を学習アルゴリズムを適用した例として，階層構造ニューラルネットワークで制御対象の同定を行なった例を示す．制御対象の状態方程式は 2.3.5節において用いた (2.32) をサンプリング幅 0.01 で離散化したものとする．学習の評価関数 Jとしては次のような誤差二乗関数を選ぶ．

$$J = \frac{1}{2}\sum_{t=1}^{500}\left[x_1^2(t)+x_2^2(t)\right]^2 \tag{2.46}$$

ここでは 3 層構造のニューラルネットワークを用い，中間層のユニット数は 10 個であるとした．図 2.21に誤差の変化を示す．また，比較のために誤差逆伝播法による学習による誤差の変化も同時に示す．ただし，誤差逆伝播法による学習では一次元探索を併用した．図 2.21より明らかなように，Powell の共役方向法に基づく学習アルゴリズムの学習速度は，誤差逆伝播法と比較するとかなり高速である．

Powell の共役方向法を学習アルゴリズムとして用いると，ネットワークの構造，制御対象や学習目的などが変更されても学習アルゴリズム自体には変更を加える必要がなく，多少の変更を行うだけでさまざまな目的の学習に用いることができ，学習を行なう際の手間が大幅に削減される．実際に，本節で例で示した制御対象の同定で用いた学習アルゴリズムのソースプログラムと制御系の学習に用いるソースプログラムと全く同一である．これに対して，勾配法を学習アルゴリズムに用いる場合には，学習アルゴリズムを同定用と制御系の学習用と区別して用意する必要がある．また，勾配法を学習アルゴリズムに用いる場合には，制御対象のヤコビアンを学習アルゴリズムに含む必要があるが．Powell の共役方向法を学習アルゴリズム

では制御対象の状態方程式，あるいは対象の数学モデルいずれも学習アルゴリズムに組む込む必要がない．よって，制御対象の詳しい情報がなくても学習アルゴリズムを作成することができる．また，制御対象が未知であっても実際に制御対象を繰り返し稼働することにより学習したり，あるいは制御対象のシミュレータに学習アルゴリズムを組み込むことが容易である．さらに，勾配法に基づく学習アルゴリズムと異なり，全ての関数が微分可能でなければならないという条件は不要であり，滑らかでない評価関数を扱うことができるほか，テーブル形式で値が得られているパラメータを持つ制御対象にも適用することができる．このために，実際の制御系設計に適する点が多い学習法であるといえるだろう．

2.4.4 計算例

本節では，Powellの共役方向法に基づく学習アルゴリズムにより最適フィードバック制御系を構築した例を示す．

制御対象 P1

制御対象を2.3.5節で用いた(2.25)であるとする．また，サンプリング幅hも同じ0.01であるとする．評価関数と制御対象の初期状態も同様に，それぞれ(2.27)，(2.28)とした．入力層と出力層の活性化関数は恒等関数，中間層の活性化関数は双曲正接関数tanhとし，中間層のユニットの数は15個であるとした．ここでも2.3.5節と同様に定常最適レギュレータによる制御との比較を行なう．図2.22に制御入力を，図2.23にそのときの状態変数の推移を示す．これらの図より，Powellの共役方向法に基づく学習アルゴリズムによって，ニューラルネットワークは最適レギュレータを学習していることが分かる．評価関数は最適レギュレータでは4.040×10^3であったが，ニューラルネットワークの学習により構築された制御系でも4.040×10^3と等しくなった．また，2.3.5節と同様に，制御対象の初期状態を領域(2.29)内で変化させたときの評価関数(2.27)を調べることにより，学習済みのニューラルネットワークの汎化性を調べる．図2.24に，領域(2.29)におけるニューラルネットワークによる評価関数と最適レギュレータによる評価関数の比を示す．図2.24より，領域(2.29)の中では最適レギュレータからの誤差が1%未満となっていることが分かり，勾配法に基づく学習アルゴリズムにより学習を行なった場合と同様に十分な汎化能力を有していることが分かった．

勾配法に基づく学習アルゴリズムでは，微分可能でない評価関数を用いることができないため，状態拘束を考えるために不連続なペナルティ項を付け加えることができなかった．このため，評価関数を(2.30)のように緩やかな拘束をつけたx_2の絶対値が1を超えない制御系を求めた．しかし，図2.11を見れば分かるように，連続的に変化する緩やかな拘束条件を付加するだけでは学習の結果得られた制御系ではx_2の絶対値は1を超えてしまい，拘束条件を破る解を学習してしまった．実際に，勾配法による学習アルゴリズム

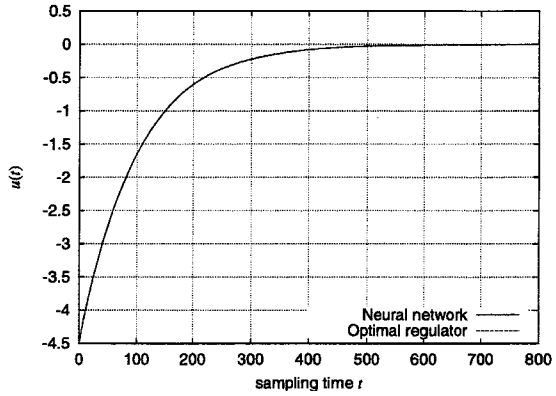


図 2.22: Comparison between a trained neural network and by a optimal regulator(Plant P1)

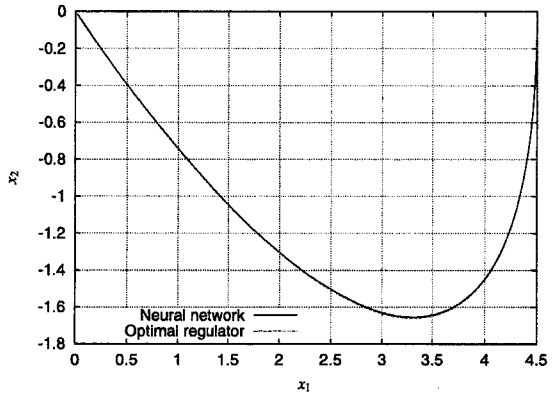


図 2.23: Transtion of the state(Plant P1)

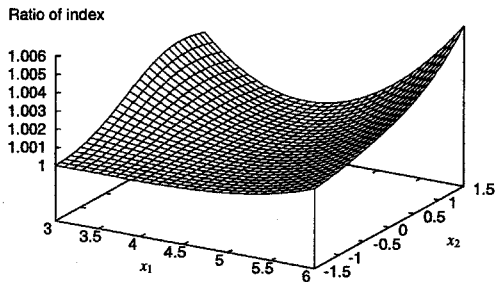


図 2.24: Performance indices from various initial states(Plant P1)

では評価関数の x_2の項の次数を ∞ にしない限り，x_2の絶対値が 1 を超えないようにすることはできない．これに対して，Powell の共役方向法に基づく学習アルゴリズムでは，評価関数の微分可能性を必要としないので，不連続なペナルティ項を扱うことができ，ペナルティ法により拘束条件を学習に組み込むことが容易となる．そこで，(2.47) のよう不連続なペナルティ項を付加した評価関数を学習に用いて制御系を構築を行なった例を示す．ただし，x_2に加えた拘束条件のために制御対象の反応が遅くなるので，原点へ収束するまでに要する時間が長くなる．このために，サンプリング幅 h を同一にすると学習における制御終端時間も長くする必要が生じるが，これは計算時間が増大を意味する．今回は計算時間を短縮するために，サンプリング幅 h を 0.03 と変更した．学習に用いた評価関数は，

$$J = \sum_{t=1}^{500} \left[x_1^2(t) + \phi(x_2(t)) + u^2(t-1) \right] \tag{2.47}$$

であり，$\phi(x)$ は以下のような不連続な関数である．

$$\phi(x) = \begin{cases} x^2 & |x| \leq 1 \\ 10^5 & |x| > 1 \end{cases} \tag{2.48}$$

ただし，制御対象の初期状態は (2.28) と同じであるとし，評価関数を (2.27) とおいたときと全く同じ構造のニューラルネットワークを用いて学習を行った．図 2.25に制御入力，図 2.26に状態変数の推移を示す．図 2.26より，$x_2 = -1$ に接するような制御結果が得られているが，これは x_2の拘束が十分に厳しいためである．また，評価関数の値は 1.481×10^3であったが，評価関数の値からも x_2の絶対値が 1 を越えなかったことが分かる．さらに，学習済みのニューラルネットワークの汎化能力を確かめるために初期状態のみを変更した場合の状態変数の推移を調べた．初期状態を $x_1(0) = 6.0, 5.0, 4.0, 3.0$ と変更したときの状態変数の推移を図 2.27に示す．この図によると，学習を行った初期状態より大きな $x_1(0)$ から始まった状態遷移軌道では，x_2の絶対値が 1 を越えてしまっていることが分かる．これは学習の際にニューラルネットワークが経験していない状態空間では，制御系がうまく学習できていないためであると考えられる．そこで，次の初期状態から学習をしたニューラルネットワークを用いて初期状態を変更した場合の状態変数の推移を調べる．

$$x_1(0) = 6.0\ ,\ x_2(0) = 0.0 \tag{2.49}$$

学習が完了したニューラルネットワークにより制御を行ったときの状態変数の推移を図 2.28に示す．この結果より，学習を行った初期状態より小さな $x_1(0)$ から出発する状態遷移軌道では $|x_2|$ は 1 を越えず，ニューラルネットワークが学習の際に経験していない状態空間の領域において汎化能力が備わっていることが期待できず，実際に用いる状態を含んだ領域を学習の際に経験する必要があると考えられる．

制御対象 P2

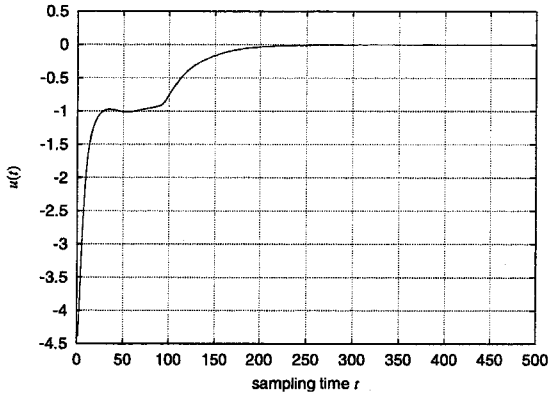


図 2.25: Control by a trained neural network(Plant P1)

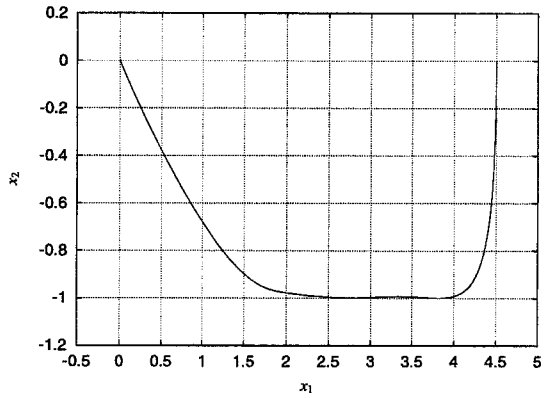


図 2.26: Transtion of the state(Plant P1)

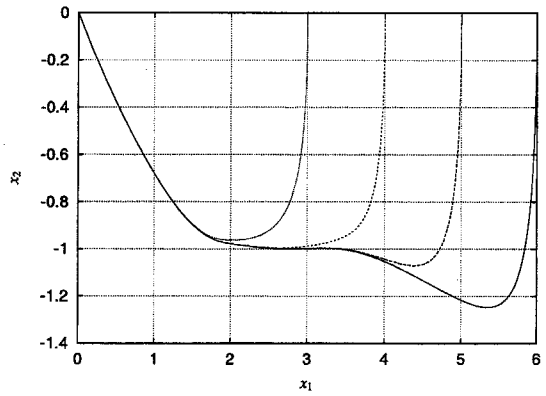


図 2.27: Trasition of the state from various initial states(Plant P1)

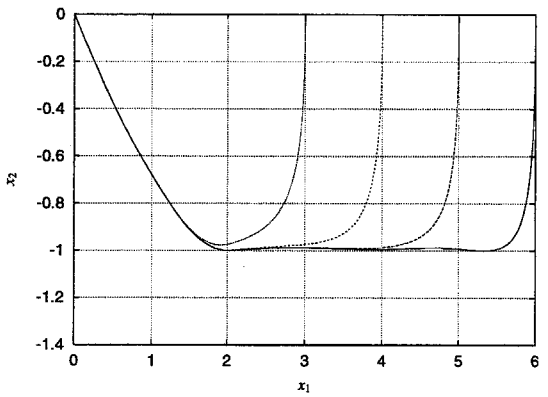


図 2.28: Trasition of the state from various initial states(Plant P1)

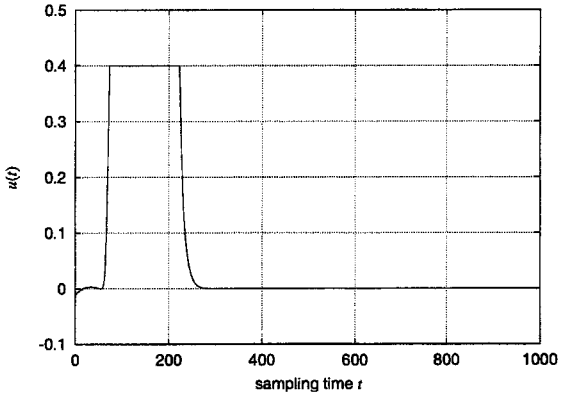


図 2.29: Optimal feedback control(Plant P2)

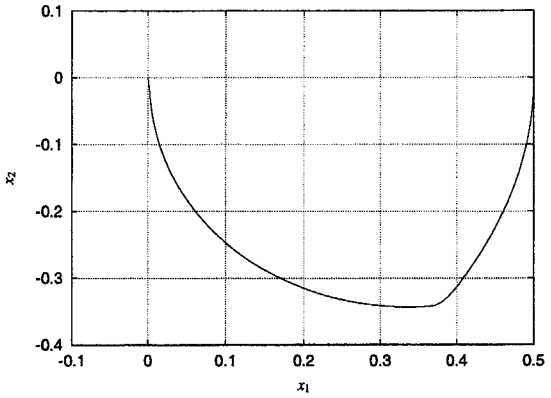


図 2.30: Transtion of the state(Plant P2)

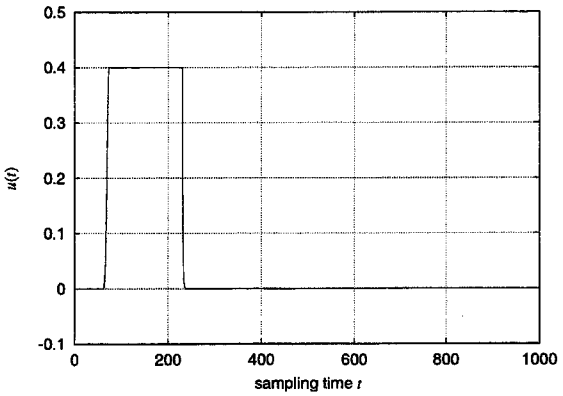


図 2.31: Optimal feedback control with dead zone(Plant P2)

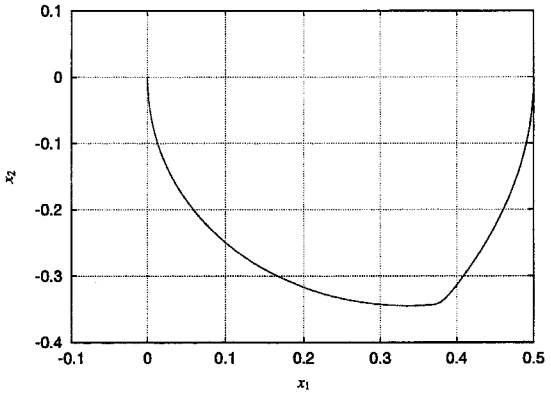


図 2.32: Transtion of the state with dead zone(Plant P2)

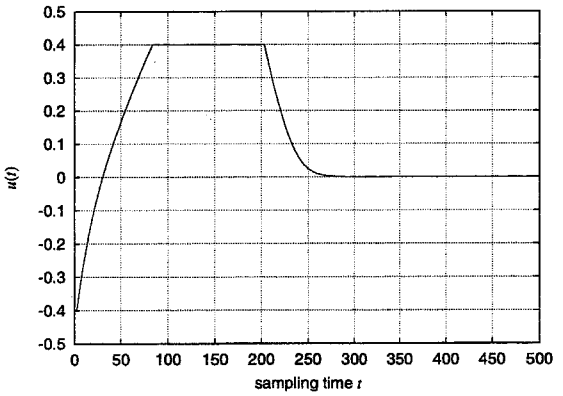


図 2.33: Optimal feedback control(Plant P2)

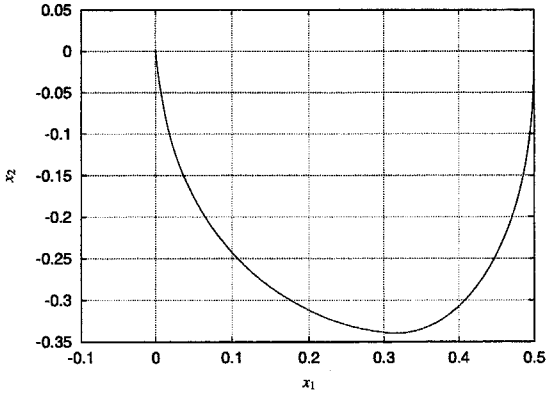


図 2.34: Transtion of the state(Plant P2)

2.3.5節で用いた (2.32) をサンプリング幅を 0.01 で Euler 差分をとったものを制御対象とする．ただし，入力には制約条件 (2.50) が存在するとする．

$$|u(t)| \leq 0.4 \tag{2.50}$$

(2.51) の評価関数を最小にするフィードバック制御系の構築を考える．

$$J = \sum_{t=1}^{1000} \left[x_1^2 + x_2^2 + |u(t-1)| \right] \tag{2.51}$$

しかし，評価関数 (2.51) は，$u(t)$ に関して微分不可能であるため，勾配法に基づく学習アルゴリズムにより学習を行なうことはできない．また，この場合は最適フィードバック解を解析的に得ることはやはり困難であるが，Powell の共役方向法に基づく学習アルゴリズムを用いることにより学習を行うことが可能となる．ここではニューラルネットワークの中間層に含まれるユニットの数を 15 個とした．出力層の活性化関数には (2.52) を用い，中間層の活性化関数は双曲正接関数を用いた．

$$F_O(x) = \begin{cases} x & |x| \leq 0.4 \\ 0.4 \cdot sgn(x) & |x| > 0.4 \end{cases} \tag{2.52}$$

また，学習では制御対象の初期状態を以下のように設定した．

$$x_1(0) = 0.5 \ , \ x_2(0) = 0.0 \tag{2.53}$$

図 2.29にニューラルネットワークによる制御入力，図 2.30にこのときの状態変数の推移を示す．このとき，学習の結果，評価関数の値は 8.529×10^1であった．最大値原理によると，(2.51) のような評価関数における最適制御入力は，0 と最大値，あるいは最小値との切り換えとなる．しかし，図 2.29より，学習により得られた制御系では制御入力は 0 と最大制御量との完全な切り換えとはなっていないことが分かる．そこで，出力層の活性化関数のみを (2.54) のように，不感帯を有するものに変更して学習を行った．不感帯をもつ関数を用いた理由は，0 と入力最大値，あるいは最小値との切り換えを実現するためには出力層の活性化関数に不感帯を持つものを用いるのが適すると考えたためである．

$$F_O(x) = \begin{cases} x + 0.01 & -0.41 < x < -0.01 \\ 0 & |x| \leq 0.01 \\ x - 0.01 & 0.01 < x < 0.41 \\ 0.4sgn(x) & |x| \geq 0.41 \end{cases} \tag{2.54}$$

学習が完了したニューラルネットワークによる制御入力を図 2.31，図 2.32にその状態変数の推移を示す．図 2.31より，制御入力はほぼ完全な切り換えとなったことが分かる．また，このとき評価関数の値は 8.496×10^1

であったが，出力層の活性化関数を (2.54) と変更することにより，最適化が進んだことが分かる．これは，出力層の活性化関数が (2.54) を用いると，ニューラルネットワークへの入力が **0** 以外のときにネットワークの出力が完全に **0** となることは困難となるために，ニューラルネットワークの学習を行うことが難しかったためであると考えられる．

さらに，学習に用いる評価関数を (2.51) から (2.55) と変更し，同じ構造のニューラルネットワークを用いて学習を行なった．

$$J = \sum_{t=1}^{500} \{|x_1(t)| + |x_2(t)| + u^2(t)\} \tag{2.55}$$

図 2.33に学習が完了したニューラルネットワークによる制御入力を，図 2.34にそのときの状態変数の推移を示す．このときの評価関数は 1.274×10^2であった．

このように評価関数は微分不可能であっても，Powell の共役方向法に基づく学習アルゴリズムを用いることにより，ニューラルネットワークは最適フィードバック制御系を学習することができる．

2.5 最短時間制御問題への適用

2.5.1 最短時間制御問題

ここでは最短時間制御問題の定式化を行なう．ただし，取り扱う系は離散時間系であるので，正確には最短段階制御問題である．前節までと同様に，フィードフォワード解を求めるのではなく，ニューラルネットワークの学習によりフィードバック解を求めることを目的とする．

[問題 *A*]

システムの状態方程式

$$\boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2.56}$$

において初期条件，及び終端条件

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \quad : \text{既知} \tag{2.57}$$

$$R_i(\boldsymbol{x}(T)) = 0 \quad i = 1, \cdots, \nu \tag{2.58}$$

のもとで次の評価関数

$$J = T \tag{2.59}$$

を最小にするような制御入力

$$\boldsymbol{u}(t) = \boldsymbol{g}(\boldsymbol{x}(t)) \tag{2.60}$$

を求めよ．ただし，

$$\boldsymbol{u} \in \Omega \tag{2.61}$$

とする．ここで$\boldsymbol{x}(t)$はn次元状態変数ベクトル，$\boldsymbol{u}(t)$はr次元入力変数ベクトル，及び$\boldsymbol{f}$はn次元ベクトル値ベクトル関数である．また，Ωは有界凸集合である．

明らかなように，最短時間制御問題における評価関数 (2.59) は微分不可能である．よって，2.3節において示した勾配法に基づく学習アルゴリズムをそのまま用いることはできない．また，2.4節において示したPowellの共役方向法に基づく学習アルゴリズムであっても，評価関数が離散的な値となるために，評価関数を減少させることができず，学習が途中で停止してしまう可能性が高い．よって，(2.59) を直接最適化を行う方法では解くことが非常に難しいといえる．

2.5.2 学習による最短時間制御系の構築

[**問題** A] をそのまま解くことは困難であるために，次の [**問題** A_τ] を考える．

[**問題** A_τ]

システムの状態方程式

$$\boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2.62}$$

において次の初期条件

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \quad : \text{既知} \tag{2.63}$$

のもとで評価関数

$$J_\tau = \sum_{i=1}^{\nu} \{R_i(\boldsymbol{x}(\tau))\}^2 \tag{2.64}$$

を最小にするような制御入力

$$\boldsymbol{u}(t) = \boldsymbol{g}(\boldsymbol{x}(t)) \tag{2.65}$$

を求めよ．ただし，

$$\boldsymbol{u} \in \Omega \tag{2.66}$$

とする．ここで$\boldsymbol{x}(t)$はn次元状態変数ベクトル，$\boldsymbol{u}(t)$はr次元入力変数ベクトル，及び$\boldsymbol{f}$はn次元ベクトル値ベクトル関数である．また，Ωは有界凸集合である．

[**問題** A_τ] はあらかじめ決められた終端時間τにおける状態の評価J_τを最小にするフィードバック制御入力を求める問題である．[**問題** A_τ] はMayer型の評価関数をもつ最適制御問題であるが，ニューラルネットワークの学習によれば，このような問題を解くことができることは前節までにすでに示されている．

[**問題** A] を解くことにより得られた最短時間をτ_*とすると，$\tau < \tau_*$を満たすτでは

$$J_\tau > J_{\tau_*} = 0.0 \tag{2.67}$$

となることは，τ_*の最適性より明らかである．よって，τ_*はJ_τが0となる最短時間と等しい．よって，τ_*よりも小さなτから [**問題** A_τ] を解き，さらにJ_τを調べ，はじめてJ_τが0となる最短のτを求めればよいことになる．あるいは，τ_*より大きなτから [**問題** A_τ] を解き，J_τを調べ，初めて0でなくなるτを求めてもよい．すなわち，最短時間制御問題 [**問題** A] はMayer型評価関数をもつ最適制御問題の集合の要素である [**問題** A_τ] をそれぞれ解き，その問題の集合の中でτの最適化を行なうこと，すなわち二重の最適化を行なうことにより解けるということが分かる．

ここで述べた方法では，未知であるはずのτ_*より小さなτから始める，あるいはτ_*より大きなτから始めるということは不可能であるように思われるが，実際にはτ_*はおよその見当をつけることが可能である．全く見当がつかない場合には，対象とする制御系に対して十分に短い時間，あるいは長い時間から始めればよいので学習は実行可能である．入力の拘束条件に関しては，評価関数に入力の不等式拘束条件をつけることにより，制御系の学習を行うこともできるが，ニューラルネットワークの出力層の活性化関数を 2.36) のように飽和特性をもつ関数を選ぶことにより，拘束条件を必ず満足するようにニューラルネットワークの出力範囲を制限し，学習を行なう方が学習が簡単になる上に，学習時間の面からも効率が高く，実際に用いる際に拘束条件を破ってしまうこともなくなる．

ニューラルネットワークの学習は数値計算によって行なわれるが，実際の数値計算ではMayer型の評価関数が完全に0となることはほとんど不可能である．よって，あらかじめ決められたεより小さくなれば，0となったとみなすことにする．しかし，εはある値より大きく設定してしまうと，学習により得られる制御系は最短時間制御系と異なるものとなる．また，εを小さくするほど，学習により得られる制御系は最短時間制御系に近づくが，計算に必要な時間が長くなってしまうので注意が必要である．実際には制御対象が離散時間システムであるために，あるε_0が存在し，それより小さなεを用いても，学習の結果により求められる最短時間τ_*は変化しなくなる．

本節で述べたように問題を置換することにより，勾配法，あるいはPowellの共役方向法による学習アルゴリズムを用いて最短時間制御系を学習により求めることが可能となる．しかし，以下のような理由により，勾配法により学習アルゴリズムを用いる場合には注意が必要である．制御入力の拘束条件を必ず満足させるために出力層の活性化関数を飽和関数を用いるが，完全なBang-Bang制御入力をニューラルネットワークが出力するようになると，活性化関数の飽和特性のために$\partial \boldsymbol{u}(t)/\partial w$は$\mathbf{0}$となる．その結果，$\partial \boldsymbol{x}(t+1)/\partial w$も$\mathbf{0}$となり，勾配$\partial J/\partial w$は$\mathbf{0}$となってしまう．このために，勾配法に基づく学習アルゴリズムでは，任意のBang-Bang制御で学習が停止してしまうことが分かるが，[**問題** A] の解はBang-Bang解，あるいは

Bang-Bang 解に近くなることが予想される．よって，勾配法に基づく学習アルゴリズムを用いて学習を行なうときには注意が必要となる．しかし，Powell の共役方向法による学習アルゴリズムには，このような欠点は存在しないので，最短時間制御問題を解く場合には Powell の共役方向法を学習アルゴリズムとして用いる方がよい．

著者とほぼ同じ時期に，Niesler らによりニューラルネットワークにより最短時間制御問題を解く方法が提案されている [27]．Niesler らも本節で述べたように問題を変換しているが，勾配法に基づく学習アルゴリズムを学習に用いているために十分な結果が得られていなかった．

2.5.3 計算例

本節では，最短時間制御系をニューラルネットワークの学習により構築した例を示す．

制御対象 T1

制御対象の状態方程式とその初期状態を

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} 1.0 & 0.1 \\ -0.05 & 0.98 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0.0 \\ 0.1 \end{bmatrix} \begin{bmatrix} 0 \\ u(t) \end{bmatrix} \tag{2.68}$$

$$x_1(0) = 0.5\ ,\ x_2(0) = 0.0 \tag{2.69}$$

であるとする．また，入力の拘束条件を (2.70) とする．

$$|u(t)| \leq 0.2 \tag{2.70}$$

このシステムの状態を最短時間で原点に遷移させるフィードバック制御器をニューラルネットワークの学習により構築する．問題を変換した評価関数 J_τ を

$$J_\tau \ = \ \frac{1}{2}\sum_{i=1}^{2} x_i^2(\tau) \tag{2.71}$$

とし，計算の打ち切り範囲を示すεは 1.0×10^{-8}であるとした．また，ニューラルネットワークの中間層に含まれるユニットの数は 20 個とした．入力の拘束条件 (2.70) を満たすために，出力層の活性化関数には，

$$F_O(x) \ = \ \begin{cases} x & |x| \ < \ 0.2 \\ 0.2sgn(x) & |x| \ \geq \ 0.2 \end{cases} \tag{2.72}$$

を用いた．学習には Powell の共役方向法に基づく学習アルゴリズムを用いたところ，J_τをϵ以下にする最小のτは 28 という結果が得られた．最短時間制御入力を図 2.35にそのときの状態変数の推移を図 2.36に示す．これら図を見ると最短時間制御入力は Bang-Bang 制御入力にかなり近くなっていることが分かる．また，図 2.36には入力の切り換え曲線がとられられている．

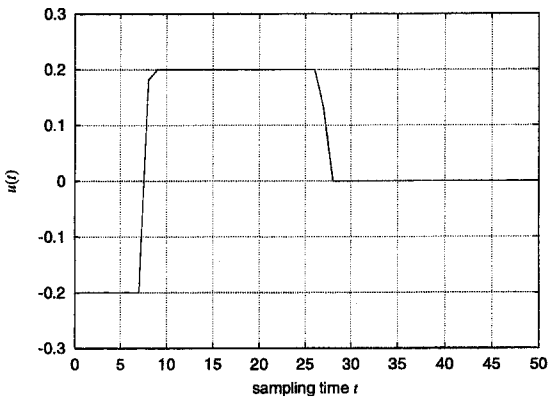


図 2.35: Time optimal feedback control(Plant T1)

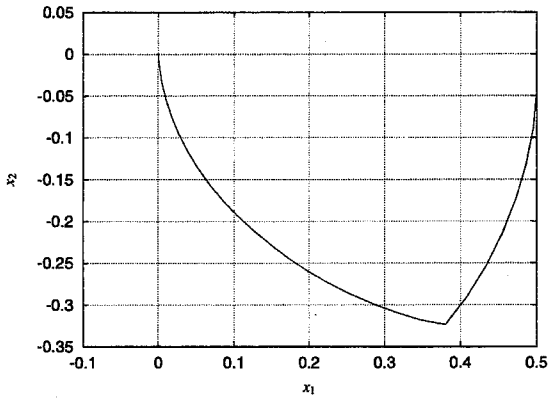


図 2.36: Transtion of the state(Plant T1)

制御対象 T2

制御対象を 2.3.5節で用いた (2.32) をサンプリング幅を 0.01 とし Euler 差分をとったものとする．制御対象の初期状態は

$$x_1(0) = 0.5\ ,\ x_2(0) = 0.0 \tag{2.73}$$

であるとする．また，入力の拘束条件を (2.74) とする．

$$|u(t)| \leq 0.4 \tag{2.74}$$

であるとする．このシステムの状態を最短時間で原点に遷移させるフィードバック制御器をニューラルネットワークの学習により構築する．問題を変換した評価関数 J_τ を

$$J_\tau \ = \ \frac{1}{2}\sum_{i=1}^{2} x_i^2(\tau) \tag{2.75}$$

であるとし，計算の打ち切り範囲を示すεは 1.0×10^{-8}であるとした．また，ニューラルネットワークの中間層に含まれるユニットの数は 15 個とした．入力の拘束条件 (2.70) を満たすために，出力層の活性化関数としては 2.4.4節で用いた (2.52) を，学習には Powell の共役方向法に基づく学習アルゴリズムを用いたところ，J_τをϵ以下にする最小のτは 217 となった．これは実時間に換算すると 2.17 秒である．最短時間制御入力を図 2.37に，そのときの状態変数の推移を図 2.38に示す．

この制御対象は非線形な対象であるために，線形な制御対象のように切り換え回数が (次数-1) 回とはならない可能性がある．この例を示すために，制御対象の初期状態を

$$x_1(0) = 1.0\ ,\ x_2(0) = 0.0 \tag{2.76}$$

と変更し，学習を行なった．ただし，計算時間を縮小するためにサンプリング幅を 0.05 とした．このときの最短時間はサンプリングタイム数で 102 となった．これは実時間に換算すると 5.10 秒である．図 2.39に

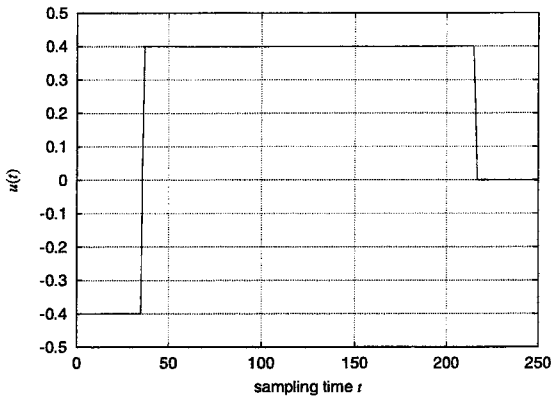


図 2.37: Time optimal feedback control(Plant T2)

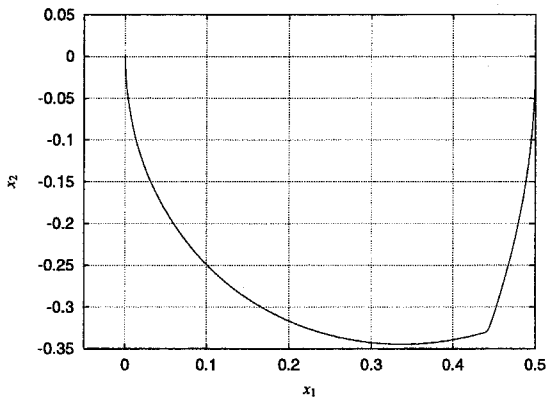


図 2.38: Transtion of the state(Plant T2)

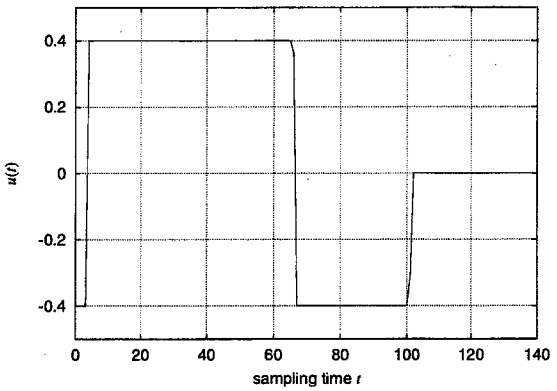


図 2.39: Time optimal feedback control(Plant T2)

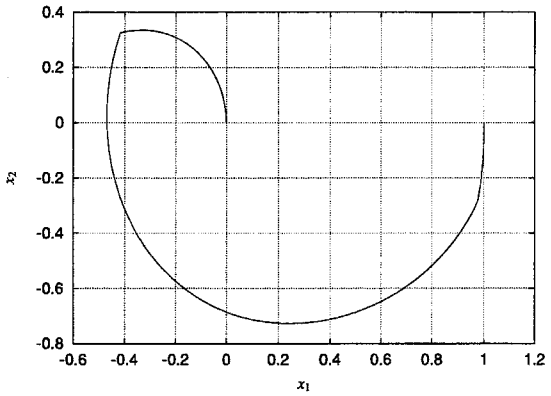


図 2.40: Transtion of the state(Plant T2)

最短時間制御入力を，図 2.40にそのときの状態変数の遷移を示す．図 2.39より，この制御対象では制御入力の切り換えが 2 回必要であったことが分かる．このように非線形な制御対象であっても，線形な制御対象と同様に最短時間制御問題の解が，しかもフィードバック制御系として求められる．

制御対象 T3

制御対象の方程式を 2.3.5節の (2.39) をサンプリング幅 0.05 で Euler 差分したものとする．制御対象の初期状態は

$$x_1(0) = 0.5 \ , \ x_2(0) = x_3(0) = x_4(0) = 0.0 \tag{2.77}$$

とする．また，入力の拘束条件を (2.78) とする．

$$|u(t)| \leq 0.4 \tag{2.78}$$

このシステムの状態を最短時間で原点に遷移させるフィードバック制御器をニューラルネットワークの学習により構築する．問題を変換した評価関数 J_τ を

$$J_\tau = \frac{1}{2}\sum_{i=1}^{4} x_i^2(\tau) \tag{2.79}$$

であるとし，計算の打ち切り範囲を示す ε は 1.0×10^{-8} であるとした．また，ニューラルネットワークの中間層に含まれるユニットの数は 15 個とした．入力の拘束条件 (2.70) を満たすために，出力層の活性化関数として (2.52) と同じものを用い．ここでも学習には Powell の共役方向法に基づく学習アルゴリズムを用いた．この制御対象の次数は 4 であるが，一般に次数が 2 を超えてしまうと位相面解析が使えず，制御入力が切り換え曲面で切り換わるようになるために最適解を得ることが難しい．しかし，本節の手法によると，フィードバック解を求めることが容易となる．学習の結果，この制御対象を原点に移動させる最短時間はサンプリングタイム数で 96 であった．これを実時間に換算すれば 4.80 秒となる．図 2.41に最短時間制御入力を示す．また，図 2.42，2.43にはそのときの状態変数の推移を示す．

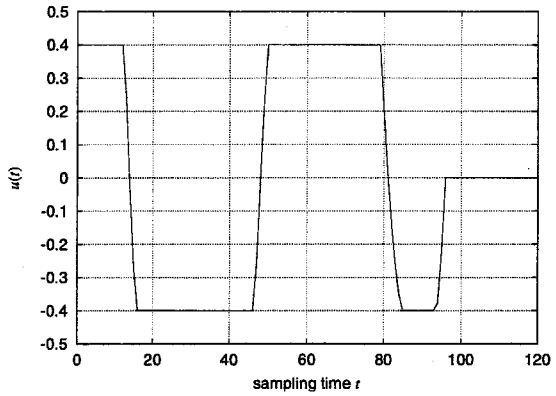


図 2.41: Time optimal feedback control(Plant T3)

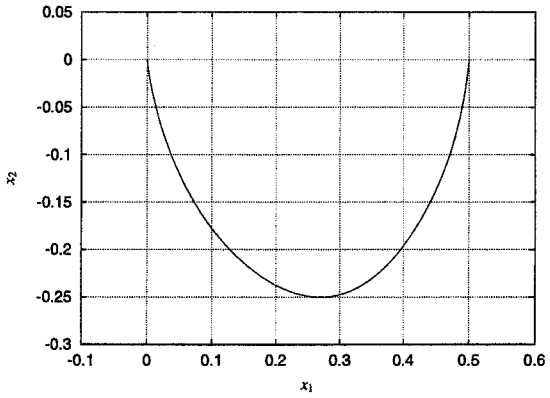


図 2.42: Transtion of the state(Plant T3)

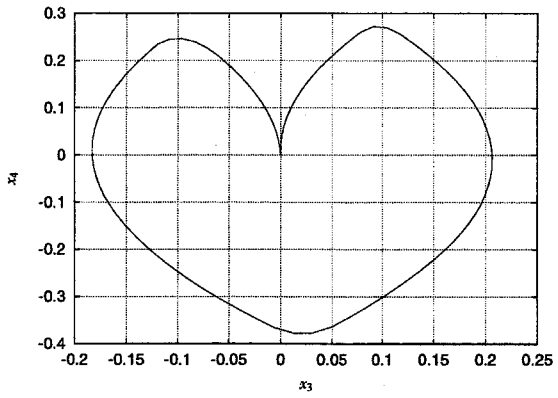


図 2.43: Transtion of the state(Plant T3)

2.6 一定目標値への追従制御問題

2.6.1 問題の定式化

2.2節において定義した問題は目標値が原点であるレギュレータ問題であったが，ここでは目標値が原点以外である場合を考える．次のように問題を定式化する．

[問題 S]

システムの状態方程式と出力方程式がそれぞれ (2.80)，(2.81) と表されている系を考える．

$$\boldsymbol{x}(t+1) \;=\; \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2.80}$$

$$\boldsymbol{y}(t) \;=\; \boldsymbol{h}(\boldsymbol{x}(t)) \tag{2.81}$$

ここでは，初期条件を

$$\boldsymbol{x}(0) \;=\; \boldsymbol{x}_0 \tag{2.82}$$

とする．出力$\boldsymbol{y}(t)$をあるステップ状の関数$\boldsymbol{r}(t)$に追従させる問題を考える．$\boldsymbol{y}(t) = \boldsymbol{r}(t)$となったときの定常入力と定常状態をそれぞれ$\boldsymbol{U}_D$及び$\boldsymbol{x}_D$として，評価関数を

$$J \;=\; \sum_{t=1}^{T} J_0(\boldsymbol{x}_D - \boldsymbol{x}(t), \boldsymbol{U}_D - \boldsymbol{u}(t)) \tag{2.83}$$

とする．ただし，ここで J_0はスカラー値非負関数である．この評価関数 Jを最小にするような制御入力を求めよ．

2.6.2 ニューラルネットワークの学習による制御系構築法

[問題 S] においても，これまで同様にフィードバック解を求めることを考える．ニューラルネットワークの学習により [問題 S] を解く方法として，二つの方法が考えられる．第一の方法は，サーボシステムの利用である．一定目標値に追従するために必要な次数の積分要素をコントローラに含ませる方法である．この方法では制御対象の次数が増えてしまうものの，前節までに示した方法をそのまま利用できるので，学習方法としては変更する点がない．

第二の方法は，フィードフォワード制御により$\boldsymbol{U}_D$を加えるものとして$\boldsymbol{U}_D$からの変化量をニューラルネットワークに出力させるものである．この方法では，2.2節と同じように，$\mathbf{0}$-入力のときの出力は必ず$\mathbf{0}$であるニューラルネットワークが必要になる．制御系のブロック線図を図 2.44に示す．勾配法，Powell の共役方向法に基づく学習アルゴリズムいずれもこの問題を解くために適用可能である．特に，Powell の共役方向法に基づく学習アルゴリズムにおいては，学習プログラムを全く変更する必要がないのに対して．勾

配法に基づく学習アルゴリズムではネットワークへの入力が$\boldsymbol{x}_D - \boldsymbol{x}$に変わったことにより若干アルゴリズムの修正が必要である．ここでは，勾配法に基づく学習アルゴリズムに必要である勾配$\partial J/\partial w$の計算式を導く．$\partial J/\partial w$と$\partial \boldsymbol{x}(t)/\partial w$ は 2.3節における計算式 (2.10)，(2.14)，(2.15)，(2.16) と同じである．2.3節と違いが生じるのは，$\partial \boldsymbol{u}(t)/\partial w$ である．これらは，

$$\frac{\partial u_i(t)}{\partial \alpha_{pq}} = F'_O(net_i^O)\left\{\beta_{ip}F'_H(net_p^H)I_q(t) - \sum_{j=1}^{N}\beta_{ij}F'_H(net_j^H)\sum_{k=1}^{n}\alpha_{jk}F'_I(x_{Dk}-x_k(t))\frac{\partial x_k(t)}{\partial \alpha_{pq}}\right\} \quad (2.84)$$

$$\frac{\partial u_i(t)}{\partial \beta_{pq}} = F'_O(net_i^O)\left\{\delta_{ip}H_q(t) - \sum_{j=1}^{N}\beta_{ij}F'_H(net_j^H)\sum_{k=1}^{n}\alpha_{jk}F'_I(x_{Dk}-x_k(t))\frac{\partial x_k(t)}{\partial \beta_{pq}}\right\} \quad (2.85)$$

$$\frac{\partial u_i(t)}{\partial \theta_p} = F'_O(net_i^O)\left\{\beta_{ip}\left(F'_H(net_p^H)-F'_H(\theta_p)\right) - \sum_{j=1}^{N}\beta_{ij}F'_H(net_j^H)\sum_{k=1}^{n}\alpha_{jk}F'_I(x_{Dk}-x_k(t))\frac{\partial x_k(t)}{\partial \theta_p}\right\} \quad (2.86)$$

のように変化する．

しかし，第二の方法では$\boldsymbol{U}_D$が既知でなければならないが，これが問題になることがある．それは，制御対象の特性を学習したニューラルネットワークを用いる場合には$\boldsymbol{U}_D$は求めることが困難であることや，制御対象の方程式が未知である場合には$\boldsymbol{U}_D$を知ることができないからである．そこで，$\boldsymbol{U}_D$の代りにその推定値である$\boldsymbol{U}_E$を用いて，推定評価関数 J_Eを

$$J_E = \sum_{t=1}^{T} L(\boldsymbol{x}_D - \boldsymbol{x}(t), \boldsymbol{U}_E - \boldsymbol{u}(t-1)) \quad (2.87)$$

と定義して，この評価関数に基づいて学習を行なう方法が考えられる．ここで，$\boldsymbol{U}_E$は$\boldsymbol{x} = \boldsymbol{x}_D$としたときのニューラルネットワークの出力である．

$$U_{Ei} = F_O(\sum_{j=1}^{N}\beta_{ij}(F_H(\sum_{k=1}^{n}\alpha_{jk}\ F_I(x_k(t)) + \theta_i) - F_H(\theta_j))) \quad (2.88)$$

学習を開始したときには，$\boldsymbol{U}_E$と$\boldsymbol{U}_D$とは異なっているが，推定評価を小さくするために$\boldsymbol{U}_D$に近づくように学習が進む．推定評価関数 J_Eを用いる学習に，勾配法に基づく学習アルゴリズムを用いるためには，(2.87) の勾配を計算しなければならない．また，それに伴って学習アルゴリズムのプログラムソースは変更する必要がある．しかし，Powell の共役方向法に基づく学習アルゴリズムに用いる場合には，学習の評価関数を (2.87) に変更するだけでよいので，少ない労力で学習を行なうことができる．

2.6.3 計算例

本節では，さまざまな制御対象に対して，ニューラルネットワークの学習により目標値追従制御系を構築した例を示す．

制御対象 S1

制御対象を2.3.5節で用いた(2.32)をサンプリング幅を0.01としEuler差分をとったものとする．ここで，出力方程式を

$$y(t) = x_1(t) \tag{2.89}$$

であるとし，制御対象の初期状態は原点であるとする．目標出力値を0.5であるとすると，定常入力及び定常状態は

$$U_D = 0.5\ ,\ x_{D1} = 0.5\ ,\ x_{D2} = 0.0 \tag{2.90}$$

となる．このような系に対して，評価関数を(2.91)のように設定し，ニューラルネットワークの学習を行なった．

$$J\ =\ \frac{1}{2}\sum_{t=1}^{1000}\left\{\sum_{i=1}^{2}(x_i(t)-x_{Di})^2\ +\ (u(t-1)-U_D)^2\right\} \tag{2.91}$$

ここではニューラルネットワークの中間ユニットは20個であるとした．勾配法に基づく学習アルゴリズムを学習に用いたところ，評価関数は2.652×10^1となった．図2.45に制御入力を示し，図2.46にそのときの出力の推移を示す．図2.46には比較のためにフィードバック制御系であるニューラルネットワークなしでフィードフォワード制御のみを行なった場合の出力も示す．ここではフィードフォワード制御入力は一定であるが，フィードフォワード制御のみでは制御対象は定常状態にならず，あるリミットサイクルに収束した．ニューラルネットワークの制御により，さまざまな目標値への追従を行なった制御結果を図2.47に示す．

さらに，制御対象の状態方程式が未知であり，それに伴いU_Dも未知であるとする．このために，次のような推定評価関数を設定して学習を行なった．(2.92)においてTは終端時刻であり，ここでは1500と設定した．

$$J=\sum_{t=1}^{T}\left[\sum_{i=1}^{2}(x_i(t)-x_{Di})^2+(u(t-1)-U_E)^2\right]+2.0\times10^5\cdot\left[(x_1(T)-x_{D1})^2+(x_2(T)-x_{D2})^2\right] \tag{2.92}$$

(2.92)はBolza型の評価関数であり，終端時間における状態の評価が加わっているが，これは学習によりU_EがU_Dが収束するのを促進するためである．ここでは制御対象が未知であるので，学習には勾配法に基づく学習アルゴリズムを用いることができないので，Powellの共役方向法に基づく学習アルゴリズムを用いた．図2.49に学習によるU_Eの遷移を示す．初期ネットワークを乱数により決定したために，学習の初期段階ではU_Dと異なっているが，学習の繰り返し回数が多くなるにつれて，$U_E = U_D$をニューラルネットワークが学習していることが分かる．U_EとU_Dの誤差は7回目の繰り返し以後は1%未満であり，十分に学習を行なったネットワークではその誤差は0.3%程度であった．学習が完了したニューラルネットワークを用いて制御を行なった結果を図2.50に示す．U_Dを既知として学習を行なったニューラルネットワークによる制御結果と完全には一致してはいないが，十分な制御結果を得ることができていることが分かる．

制御対象 S2

制御対象の方程式を 2.3.5節の (2.39) をサンプリング幅 0.02 で Euler 差分したものとする．出力方程式は

$$y(t) = x_1(t) \tag{2.93}$$

であるとし，制御対象の初期状態は原点であるとした．目標出力値を 0.5 であるとすると，定常入力及び定常状態は

$$U_D = 0.0\ ,\ x_{D1} = 0.5\ ,\ x_{D2} = 0.0,\ x_{D3} = 0.0,\ x_{D4} = 0.0 \tag{2.94}$$

である．このような系に評価関数を (2.51) と設定し，学習を行なった．

$$J\ =\ \frac{1}{2}\sum_{t=1}^{750}\left\{\sum_{i=1}^{4}(x_i(t) - x_{Di})^2\ +\ (u(t) - U_D)^2\right\} \tag{2.95}$$

ここでニューラルネットワークの中間ユニットは 10 個であるとし，Powell の共役方向法に基づく学習アルゴリズムを用いて学習を行なったところ，評価関数の値は 1.937×10^1 となった．図 2.51にニューラルネットワークによる制御入力を示し，図 2.52にそのときの出力の推移を示す．また，図 2.53に様々な目標値への追従の様子を示す．

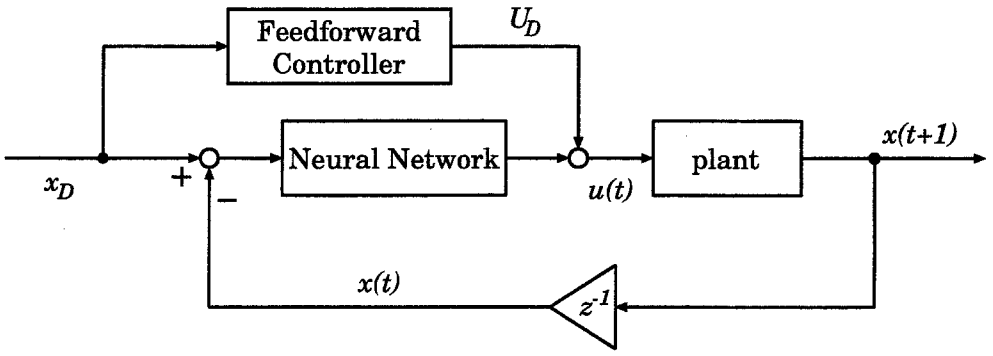


図 2.44: Block diagram for training neural network to design a trucking control system using U_D

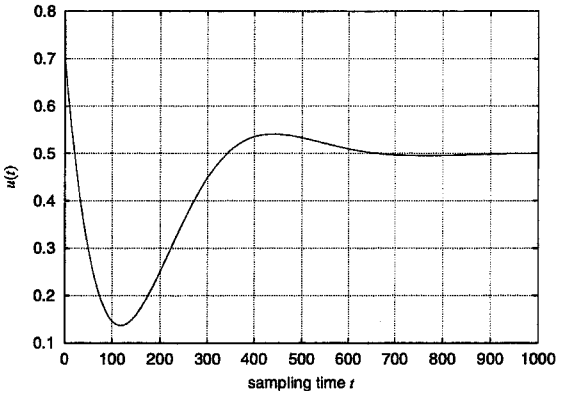


図 2.45: Control by a trained neural network(Plant S1)

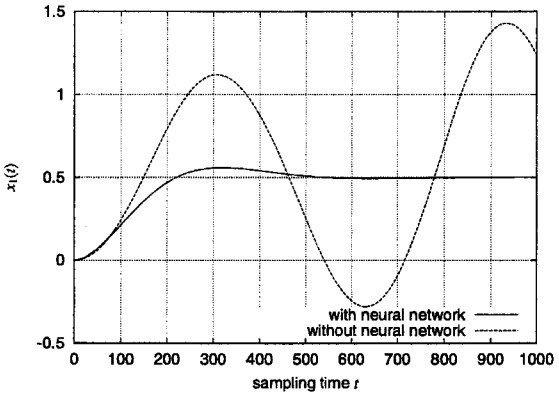


図 2.46: Trucking a desired output(Plant S1)

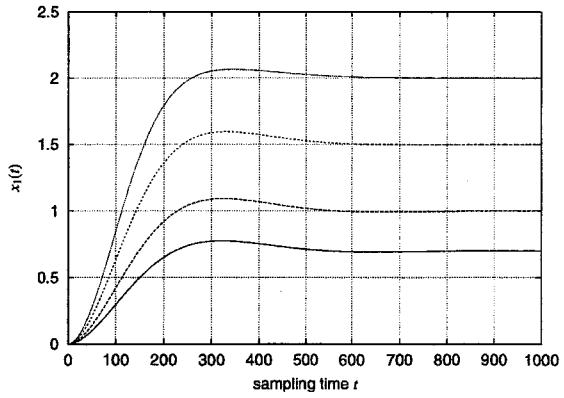


図 2.47: Trucking to various desired output(Plant S1)

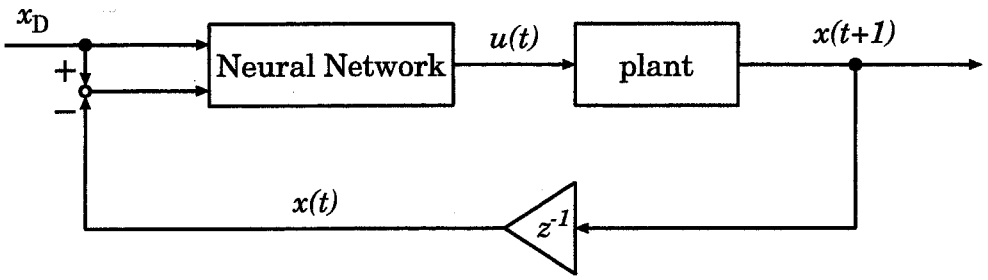


図 2.48: Block diagram for training neural network to design a trucking control system using U_E

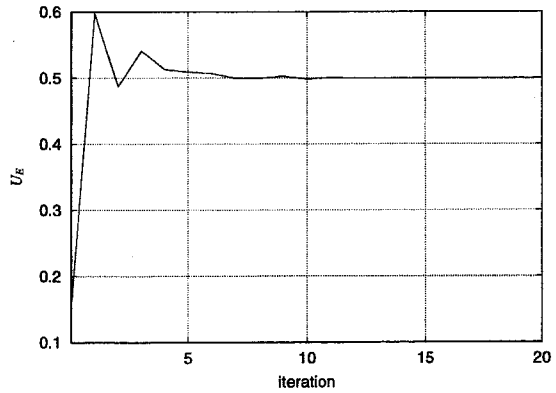


図 2.49: Transtion of U_E by training a neural network(Plant S1)

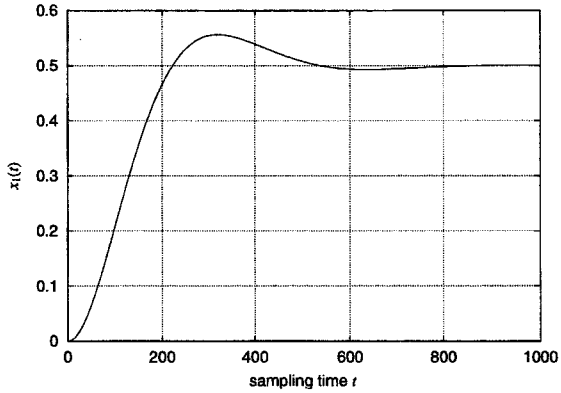


図 2.50: Trucking a desired output(Plant S1)

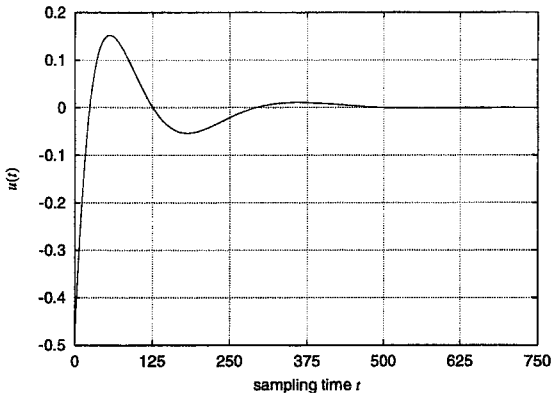


図 2.51: Control by a trained neural network(Plant S2)

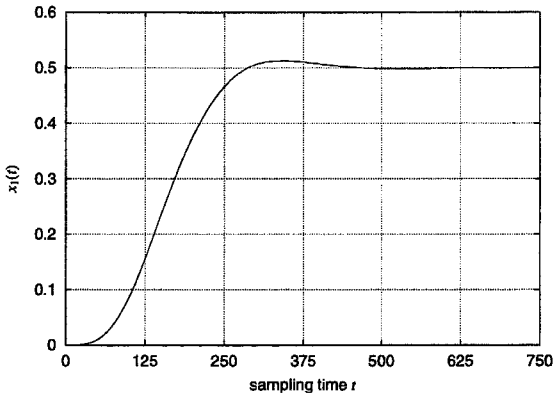


図 2.52: Trucking a desired output(Plant S2)

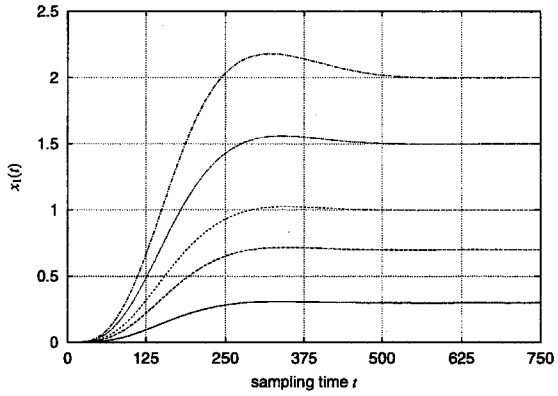


図 2.53: Trucking to various desired output(Plant S2)

2.7 逆ダイナミクス変換の学習

最適フィードバック制御系の構築とは少し異なるが，本節では制御対象の逆システムを構築する方法について考える．ニューラルネットワークを制御系に応用する研究の中で最も多く関心を集めたものは，制御対象の逆システムを構築する方法であり [5–7]，この中で最も有力だと考えられているものは，川人により提案されたフィードバック誤差学習法 [7] である．フィードバック誤差学習法では二自由度制御系となっているが，フィードフォワード制御部にニューラルネットワークを用い，フィードバック制御には適当に設計された線形制御器などを用いられているが，フィードバック制御系の設計には全く注目していない．ニューラルネットワークの学習が十分に進むと，制御対象の逆システムがニューラルネットワークにより実現される．このとき，フィードバック制御系はその役目を終えて，制御対象は主としてフィードフォワード制御によって目標軌道に追従するよう制御が行なわれると述べられている．しかし，実際にフィードバック誤差学習法により学習を行なうと，その結果はフィードバック制御系に大きく依存することが分かる．また，完全に学習が進んだニューラルネットワークであっても，フィードバック制御器を取り外してしまうと目標軌道に追従しなくなってしまう．よって，制御結果はフィードバック制御に依存しており，目標軌道への追従性の改善にのみフィードフォワード制御であるニューラルネットワークが役割を果たしている．また，逆システムを用いるために，制御対象の相対次数が 0 でなければならないことも，大きな欠点であるといえる．

このような欠点を解消するために，本節では非線形逆ダイナミクス変換 [9,10,28,29] をニューラルネットワークを用いて学習する方法を提案する．簡単のために，次のような 2 次の非線形システムを考える．

$$\ddot{x} = f(x, \dot{x}, u) \tag{2.96}$$

制御対象の目標軌道 x_cは次のモデルにより生成されるとする．

$$\ddot{x}_c = -K_D\dot{x}_c - K_P x_c + K_P r \tag{2.97}$$

(2.97) では，理想出力 rから高周波成分を取り除かれることにより目標軌道 x_cが生成されている．ここで，図 2.54に示したブロック図のような制御系を考える．この制御系では，前置補償器としてニューラルネットワークがフィードバック制御器からの擬似入力 vに変換を施すことにより，制御入力 u が生成される．ニューラルネットワークは，制御対象の非線形特性を打ち消し，入出力関係を線形化を行うために用いられる．すなわち，逆ダイナミクス変換 (2.98) をニューラルネットワークにより実現する．

$$u = f^{-1}(x, \dot{x}, v) \tag{2.98}$$

最適フィードバック制御系と異なり，目標軌道を (2.97) により求めることができるので，誤差逆伝播法を学習アルゴリズムとして学習を行う方法も考えられるが，ここでは教師信号を用いない直接的な学習法を

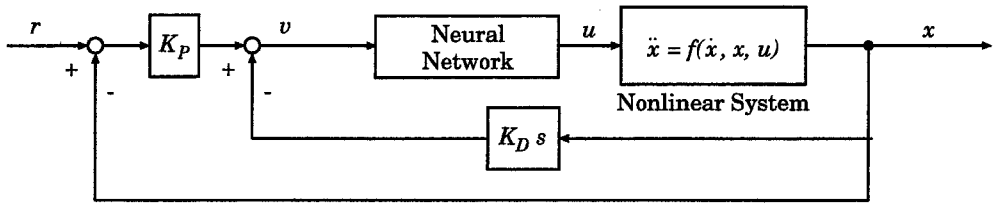


図 2.54: Block diagram of controller for linearization

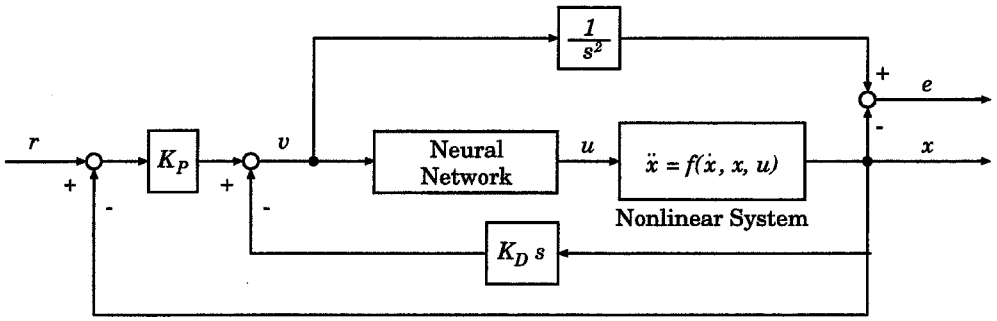


図 2.55: Block diagram for training controller for linearization

提案する．逆ダイナミクス変換 (2.98) を (2.96) に施すと，

$$\ddot{x} = v \tag{2.99}$$

となることを考慮し，図 2.55に示したブロック図をニューラルネットワークの学習に用いる．学習に用いる評価関数 Jとして，(2.100) のように積分要素からの出力と制御対象の出力との誤差二乗関数を選ぶ．

$$J = \sum_{t=0}^{T} e^2(t) \tag{2.100}$$

ここで，e は線形化誤差であり，学習の評価関数を (2.100) と選ぶことにより線形化誤差が最小化される．$J = 0$ となれば，vと x の間に (2.99) という関係が成立し，vと x の関係が線形化される．また，図 2.54 より，

$$v = K_P(r - x) - K_D\dot{x} \tag{2.101}$$

であるので，ニューラルネットワークにより逆ダイナミクスが実現されたとき，図 2.54のダイナミクスは (2.97) と一致するので，制御対象の軌道は目標軌道と一致する．評価関数 Jの勾配を求めることは，制御対象の特性 (2.96) が既知であれば可能であるが，制御対象が既知であるときには，(2.98) より制御入力 u を定めることができるので，学習を行なう意味はない．このために，(2.96) の右辺が未知である場合の学習方

法のみが必要であるが，Powellの共役方向法を学習アルゴリズムに用いることにより，制御対象の状態方程式(2.96)が未知であっても，評価関数Jを最小にするように学習を行なうことが可能である．

簡単な例として，制御対象の方程式を(2.102)と仮定し，計算を行なった例を示す．

$$\ddot{x}(t) = 0.2(1.0 - x(t)^2)\dot{x}(t) - x(t) + u(t) \tag{2.102}$$

ここでは$r = 0$として，K_PおよびK_Dは適当に定めた正の定数として学習を行なった結果を示す．図2.56に未学習のニューラルネットワークを用いた場合の制御結果を，図2.57に学習が完了したニューラルネットワークによる制御結果と線形モデル応答を示す．さらに，図2.58に原点付近の拡大図を示す．それぞれの図では複数の初期状態から制御を開始したときの解軌道を示したが，未学習のニューラルネットを用いた場合では原点に収束せず安定化されていないことが分かる．これに対して，学習が完了したニューラルネットワークにより制御された制御対象の応答はほとんど線形モデルの応答と一致しており，ニューラルネットワークの学習により入出力線形化されたことが分かる．

また，図2.59のようなブロック図において(2.100)を評価関数として学習を行なうことにより，入出力の線形化を学習する方法も考えられる．この方法で学習された制御系では学習後でも線形モデルの計算を常に行なう必要がある．これに対して，図2.54の制御系では線形モデルの計算が不要であるので，低次元な制御器が実現できる．

フィードバック誤差学習法は，人間の運動制御の学習法として提案された方法である．しかし，運動制御が全てオンライン学習により得られているとは考えにくい．もし，全ての運動の学習がオンライン学習であるとすると，運動の反復練習にはあまり意味がなく，継続的で長時間にわたる練習が運動の学習に効果を発揮することを意味する．しかし，人はスポーツなどの上達のために継続的な反復練習を行う．また，オフライン学習の効率はオンライン学習の効率に比べて高い．よって，運動制御の学習にも反復練習，すなわちオフライン学習が主に使われているのではないかと考えている．また，フィードバック誤差学習法のようにフィードフォワード制御のみに着目すると，対象の変化や雑音に弱くなってしまう．しかし，生物や人間の運動制御を見ると，変化や雑音に対して耐性の高い制御を行なっていると思われる．これは，一瞬で対象の特性変化などを学習してしまうのではなく，フィードバック制御を行なっているためであろう．

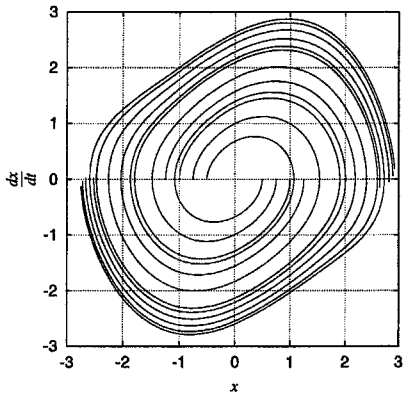


図 2.56: Response by a neural network that is not trained

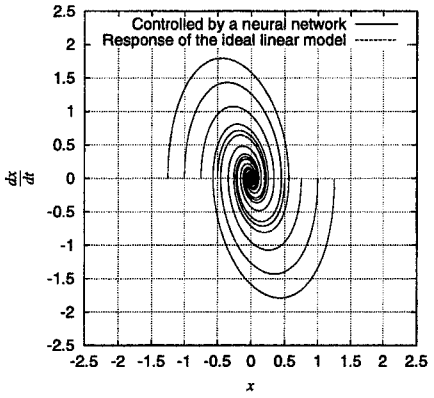


図 2.57: Response by a neural network that is trained

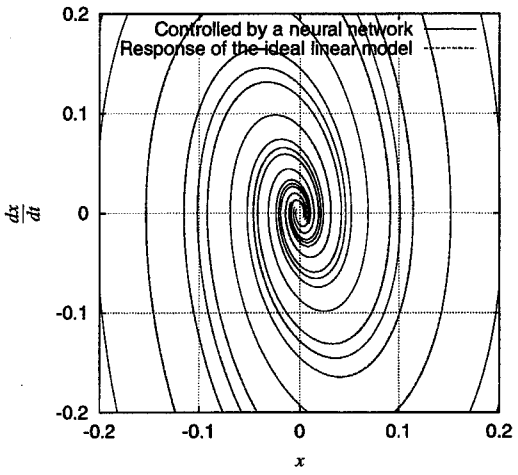


図 2.58: Response by a neural network that is trained(zooming around the origin)

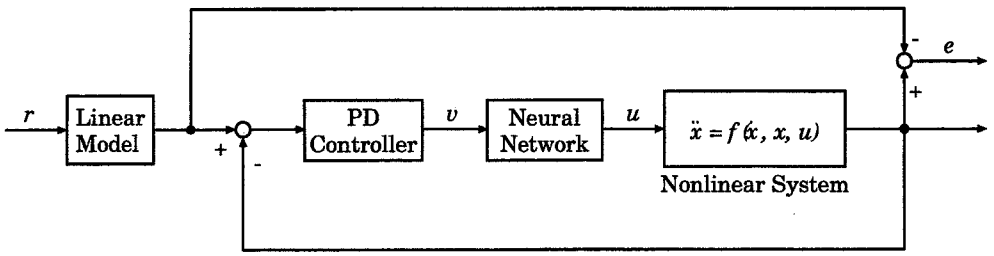


図 2.59: Block diagram for training controller for linearization including a linear model

2.8 結言

本章では，ニューラルネットワークを用いて最適フィードバック制御系を構築する手法を示した．フィードバック制御器として用いるためには，入力が**0**のときにはニューラルネットワークの出力は必ず**0**とならねばならない．一般のニューラルネットワークはこのような性質を持たないため，フィードバック制御器として用いる際に不都合が生じることが多い．特に，これは最短時間制御問題において顕著に現れる．最短時間制御問題とは，入力を積極的に加えることにより状態を最短時間で原点に遷移させる制御器を設計することを目的とする．2.2節で述べたような変更を施さず，一般の階層構造型ニューラルネットワークを学習に用いても最短時間で状態を原点に遷移させることは可能であった．しかし，学習で状態を原点に保たねばならないということは考慮されていないので，ニューラルネットワークにより最短時間で状態が原点へ遷移するが，制御入力が0とならず，状態を原点にとどめることができなかった．しかし，2.2節で述べたような変更を加えたニューラルネットワークを学習に用いたところ，最短時間制御問題においても対象の原点を安定化することが可能であった．

フィードバック制御系の学習の際に，乱数により初期化されたニューラルネットワークを用いる．これは学習の初期点を与えるために欠くことができないが，未学習であるフィードバック制御器は制御系を不安定にする可能性を持っている．しかし，数値計算が完全に発散するフィードバック制御系を学習の初期状態とすることができない．このために，定められた時間内では数値計算が完全に発散しないような制御器を初期状態として選ぶ必要がある．しかし，ニューラルネットワークの結合度の初期化には乱数を用いるので，発散しない制御器を選ぶことは難しいことがある．このような場合であっても，初期値変化法を用いることで容易に学習を行なわせることができる．初期値変化法とは，制御対象が発散してしまったならば，初期条件を各々の状態変数を1/2，またはサンプリングタイム数を半分にして問題を解き直す方法である．このように問題を変更しても発散するようならば，さらに各状態変数を1/2，またはサンプリングタイム数を半分にする．このようにして，あらかじめ学習をおこなったニューラルネットワークを初期状態とすることにより，最適なフィードバック制御系を学習することができる．実際に，本章で示したいくつかの計算例では初期値変化法を用いて学習を行っている．

本章ではフィードバック制御器の学習アルゴリズムとして，勾配法に基づく学習アルゴリズムとPowellの共役方向法に基づく学習アルゴリズムを示した．共に学習速度は高速であり，制御系設計法として十分であった．しかし，最短時間制御問題には勾配法に基づく学習アルゴリズムは適していなかった．2.5節で述べたように，勾配法に基づく学習アルゴリズムでは最適でないBang-Bang制御系であっても，その勾配は0となり学習が停止するという欠点があるからである．最適でないBang-Bang制御系に陥らない場合であっても，勾配法に基づく学習アルゴリズムにより最短時間フィードバック制御系を学習する際にはその

学習の速度は非常に遅く，一つの終端時間τに対して何千回も学習を行う必要があったが，最短時間制御はBang-Bang制御に近い制御であることが原因であると思われる．これに対して，Powellの共役方向法に基づく学習アルゴリズムを用いると，数回～数十回の学習の繰り返しだけで評価関数の減少が停滞，あるいは評価関数自体がε以下となった．さらに，多くの数値シミュレーションにおいて，勾配法に基づく学習アルゴリズムを用いて得られた解を上回った．これは勾配法に基づく学習アルゴリズムによる結果が不十分であり，学習により最適解に辿り着かなかったためと考えられる．最短時間制御問題以外の問題ではいずれのアルゴリズムにより学習を行なっても，同じように問題を解くことができるこが，Powellの共役方向法に基づく学習アルゴリズムでは制御対象の状態方程式を必要としないために勾配法に基づく学習アルゴリズムにはない利点を持つことが示された．第一の利点は，制御対象や用いるニューラルネットワークの活性則や評価関数などによりプログラムソースの変更が不要であることである．さらに，制御対象の同定に用いる場合でも制御系の学習に用いる場合でも全く同一のプログラムソースを利用できる．勾配法に基づく学習アルゴリズムでは，ソースコードの変更，それに伴うデバック作業が不可欠となるが，Powellの共役方向法に基づく学習アルゴリズムではこのような作業が全く不要となる．第二の利点は，制御対象の知識を必要とせずに学習を行なうことができることである．制御対象が繰り返し稼働できるならば，実験から学習を行なわせることも可能である．学習に要する時間から考えると，実対象を用いた学習はあまり現実的ではないかもしれないが，制御対象の動特性に関する知識を必要とせずに学習アルゴリズムを構築できるため，制御対象のシミュレータに学習アルゴリズムを組み込む際にも制御対象に関する知識を必要としない．これに対して，勾配法に基づく学習アルゴリズムでは，制御対象のヤコビアンが不可欠である．このためにシミュレータに学習を組み込もうとする際に，制御対象に対する知識が不可欠となってしまう．また，Powellの共役方向法による学習アルゴリズムは，フィードバック制御系の学習だけでなく，さまざまなニューラルネットワークの学習アルゴリズムとしても非常に有効である．特に，フィードバック制御系の学習のように，勾配を求めることが困難な相互構造型ニューラルネットワークのオフライン学習として適している．

ニューラルネットワークを制御系設計に応用する際の問題点は，そのニューラルネットワークが近似している関数が明らかでないことである．このために，ニューラルネットワークの学習により得られた制御対象のモデルを，従来の制御系の設計法で用いることは容易ではない．また，ニューラルネットワークをフィードバック制御器として用いると，制御系の安定性の解析が難しくなる．このため，本節ではニューラルネットワークの汎化能力について調べたが，制御対象の初期条件を1点に限定して学習させたにもかかわらず，かなり高い汎化能力を示すことが分かった．また，汎化能力を向上させるためには，複数の初期条件をにより学習を行うことが有効であることも分かった．残念ながら，最短時間制御問題ではニューラル

ネットワークの汎化能力は乏しかった．学習に用いた以外の初期状態であっても，状態を原点に遷移させることができるが，最短時間とはいえなくなることがことが多かった．最短時間制御問題では，二重の最適化をすることで解くために複数の初期条件を学習させることは難しいが，このような場合にニューラルネットワークの汎化能力を高めるには，間接的学習法 [24] を併用することが有効であると思われる．

さらに，制御対象に含まれる不確さに対するロバスト性が考慮されていないことも問題点といえる．最適レギュレータのロバスト性の高さより，最適フィードバック制御系を学習したニューラルネットワークのロバスト性は低くはないと予想されるが，定量的な評価が得られていない．また，学習後のニューラルネットワークの汎化能力が高いことが数値計算により確かめられたが，学習を行なった初期状態以外では最適解と多少の違いがあることが確認された．本章では，この違いを評価関数の差により評価し，ニューラルネットワークの汎化能力の指標としたが，ロバスト性にも影響を及ぼすと考えられる．汎化能力は学習を行なった最適軌道と少しずれたときを考えるが，ロバスト性は制御対象が変化したために制御結果が最適軌道とならなかったときのことを考える．よって，ロバスト性と汎化能力は密接な関係を持っていると考えられる．また，学習により得られたネットワークのロバスト性が，本来のロバスト性の大きさに比べて小さくなってしまう可能性もあるため，ロバスト性に関する定量的な評価を行なう必要がある．しかし，ニューラルネットワークを制御系に応用する他の研究では，制御対象がもつ不確さをほとんど考慮していない．不確さの考慮をしている研究では，学習により適応的に制御対象のモデル化誤差の影響を取り除くことを目的としている [9,10]．実際に，これまでの研究では不確さに対するロバスト性は定量的な評価が行なわれてだけでなく，曖昧な取り扱いしかなされていない．次の章において，この問題点を解決するためにロバスト制御系の学習方法を論じる．

第 3 章

学習によるロバスト制御系の構築

3.1 緒言

一般に制御対象の正確なモデル化は難しく，特に正確な非線形モデルを得ることは非常に困難である．正確にモデル化された対象であっても，経年変化などによって特性が変化してしまうことも多く，負荷や重量などのようにある幅で変化するパラメータも少なくない．また，工業製品では公差が存在するので，個体によってばらつきが存在することは避けられない．さらに，実際の制御対象には未知の外乱が存在し，その結果として制御結果が影響を受けることがある．このように，実際には制御対象を完全に理解し，その正確な数学モデルを作成することは困難であり，なんらかの仮定を設けることにより制御対象の数学モデルを作成する．このため．モデル化誤差をなくすことは不可能といってよい．このように，実際の制御対象には不確定な要素が含まれることが多く，このような不確定な要素を総称して制御対象の不確さという．対象に不確さが存在しても安定に制御することが可能であり，さらに制御結果の変化が小さいことが，理想的な制御系の特性といえる．このような制御系の特性を不確さに対するロバスト性という．近年，ロバスト制御系の設計法に関する研究がすすみ，H_∞制御など線形制御理論において重要な結果が導かれた [30,31]．実際の制御対象の多くは非線形性を持っているが，非線形性を無視することなどにより線形モデルを作成する．よって，モデル化誤差などの不確かさは一般に非線形性を持つが，従来のロバスト制御系の研究では線形の不確かさのみを取り扱っている．また，非線形な制御対象に対しては，複数の動作点の周りの対象の線形モデルを用いて構築した制御器を状態により切り換えるゲインスケジューリング制御が有効であるが，制御系の設計を行なう動作点をどのように選ぶのかなど制御系の設計が大変難しく，設計者の経験と勘に頼る部分が大きい．このために非線形性を直接扱うことができ，かつロバスト性を考慮することができる制御系の構築方法が必要である．

近年, 非線形な制御対象に対してニューラルネットワークを用いて制御系を構築する様々な方法が提案された [8,27,32]. これらの研究の多くは, 制御対象は完全に既知であるか, 動特性の未知部分をニューラルネットワークを用いて学習させた近似モデルに基づいて学習を行っている. しかし, 従来の研究では学習を行なったニューラルネットワークのロバスト性を考慮していなかったために, 学習に用いた制御対象のモデルと実対象の間に存在する差がどれぐらいまで許されるのか全く保証されていなかった. 第2章において示したニューラルネットワークの学習による制御系設計法ではニューラルネットワークがフィードバック制御系となることから, 制御対象の変動などに対するロバスト性をある程度は持つと期待されるが, 学習の際にそのロバスト性を定量的に保証することはできない. このために, 実際に最適フィードバック制御系を学習したニューラルネットワークのロバスト性を確認すると, 評価関数はほぼ同じとなっていても, ロバスト性が大きく異なることがあることが示された [33]. この結果は学習後のネットワークのロバスト性が期待より低くなっている事があることを意味し, 制御系のロバスト性を保証する学習方法の必要性を示唆している. ロバスト性に対する配慮を行うために, パラメータの変動が制御結果に大きな影響が現れないように, 評価関数に制御対象のパラメータ感度を重み付で加えたものを制御系の学習に用いることが考えられる. 例えば, 文献 [34] ではパラメータ変数による評価関数の微分, つまりそのパラメータの感度関数を重みをつけて評価関数に加え, 学習を行う手法を提案している. しかし, パラメータの感度を考慮する方法では, どの程度のパラメータ変動までニューラルネットワークは許容できるのか不明であったり, 評価関数に加えるパラメータの感度の重みの決定法に問題がある. また, パラメータが変化してしまったときの制御結果を考慮しないために, どの程度の制御性能の悪化が生じるのかなどは学習の段階で保証することはできない. さらに, 感度を用いる方法では, 不確かさをパラメータの不確定さに制限する必要があることも問題である.

本章では, ニューラルネットワークを用いて制御対象の不確かさに対してロバストな制御系を学習を行なう方法を提案する. 第一の方法は L_2ゲインに基づく方法で, 制御対象の非構造的な不確かさを考慮することを可能とするものである. この方法では制御系の学習と制御対象の不確かさを模擬するために, 二つのニューラルネットワークを用い, 二つのニューラルネットワークを競合するように学習させる. これにより, 制御対象の不確かさに対してロバスト性を持った制御器の学習が可能となる. また, 第一の方法では学習により得られる制御系は定量的に評価されたロバスト性を持つ. このような定量的なロバスト性の評価は他のニューラルネットワークを制御に用いる研究には見られない優位な点である. 第二の方法では, 制御対象のパラメータ変動を取り扱い, MiniMax 最適化に基づいて学習を行う. 構造的な不確さはパラメータの不確さに帰着することができるので, この方法により構造的な不確さに対するロバスト制御系を構築することができる. また, この方法でも, 学習の際に仮定する不確かなパラメータの存在範囲により, 学習後

のニューラルネットワークがロバスト性を持つ範囲は定量的に保証される．さらに，本方法で学習を行ったニューラルネットワークでは，学習の際に制御性能の最悪劣化を考慮しているので，安全に用いることができる．第二の方法は第一の方式に比べると適用範囲が狭いのは自明である．しかし，ロバスト制御系においてしばしば問題となる制御系の保守性が第一の方法に比べて小さくなるという利点がある．また，これらの提案手法は複合して用いることが可能であり，制御系の学習において不確かさに関する情報を十分に活用することができる方法に発展させることが容易である．

また，非線形ロバスト制御では，スライディングモード制御が注目を集めているが，これは状態空間中に設計したある超平面 (滑り面) に状態を拘束させるように制御を行うものである．スライディングモード制御器ではある切換面において制御入力が不連続に変化するために，可変構造型制御ともいわれ，高度なロバスト性を有することが知られている [35–37]．しかし，従来の方法では制御入力の大きさの制限を考えていないために，滑り面に状態を拘束するには大きな制御入力が必要となることが多く，入力の大きさに厳しい制限のある対象では状態を滑り面に拘束できなくなるという難点があった [38]．本研究では，最適制御問題の特異解 [39] に注目しニューラルネットワークに特異解を学習させることにより大きなロバスト性を持つ制御系が学習できることを示す．本手法により学習される制御系はスライディングモード制御系となり，従来法では困難であった入力の飽和特性を容易に取り扱うことが可能となる．また，入力の利用に厳しい制約があるときには，線形な対象に対して線形滑り面を設計した場合であっても，一般に入力切換面は線形とならず，状態空間中で曲がっていることも示される．さらに，本手法を用いることにより，適切な境界層の設定や非線形滑り面をもつ制御系の設計も容易となることを示す．

3.2 対象とする問題

本節では対象とする問題について述べる．制御対象のモデルは，次のような離散時間システムであるとする．

$$\boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t)) \tag{3.1}$$

(3.1) において$\boldsymbol{x}(t)$ は n 次元状態変数ベクトル，$\boldsymbol{u}(t)$ は m 次元入力ベクトル，$\boldsymbol{v}(t)$ は制御対象の不確かさを表す未知な関数である．$\boldsymbol{f}$は既知である n 次ベクトル値関数であり，$\boldsymbol{v} = 0$ のとき制御対象の公称モデルとなる．(3.1) は不確かさの構造に関する情報がないときの制御対象のモデルである．これに対して，存在するあるいは考慮すべき不確かさがパラメータの不確さに限定される場合には，制御対象のモデルは

$$\boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\alpha}(t)) \tag{3.2}$$

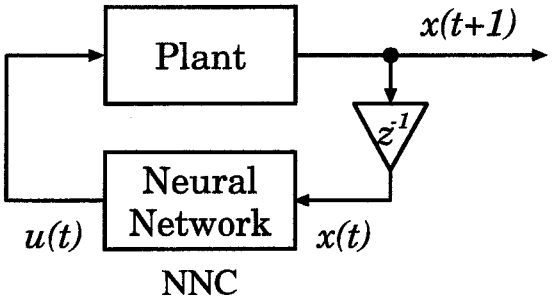


図 3.1: Block diagram of feedback controller using neural network

あるいは

$$\boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{\alpha}(t)^T \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{3.3}$$

のように書き直すことが可能である．ただし，(3.2),(3.3) において$\boldsymbol{f}$と$\boldsymbol{g}$は既知である．$\boldsymbol{\alpha}(t)$ は不確かなパラメータベクトルであり，各成分は独立であるとする．また，$\boldsymbol{\alpha}(t)$ が存在する範囲はある既知の閉集合Δ内であると仮定する．

$$\boldsymbol{\alpha}(t) \in \Delta \quad (^{\forall} t \geq 0) \tag{3.4}$$

Δはパラメータの公差などから知ることができる．ただし，曖昧な範囲でしか分からない場合には，存在すると考えられる範囲を上から覆う領域をΔと設定する．

本研究では本節で示したような不確かな制御対象に対して

$$\boldsymbol{u}(t) = \boldsymbol{u}(\boldsymbol{x}(t)) \tag{3.5}$$

のような状態フィードバック制御器としてニューラルネットワークを用いる．ただし、対象の状態は全て測定できるとし，不確かさが存在しても制御系の原点は平衡点であると仮定する．図 3.1にニューラルネットワークを用いたフィードバック制御系のブロック図を示す．ニューラルネットワークの学習によって，(3.1),(3.2),(3.3) のように不確かさが存在する制御対象の原点を安定にするロバストな制御系の構築することが目標である．

3.3 非構造的な不確かさに対するロバストな制御系

3.3.1 ニューラルネットワークによる不確かな系の表現

本節では非構造的な不確かさを含む系 (3.1) を対象とする．非構造的な不確かさとは，関数の形など構造的な情報を含まない不確かさのことをいう．このために，不確かさに関する情報がほとんどない場合で

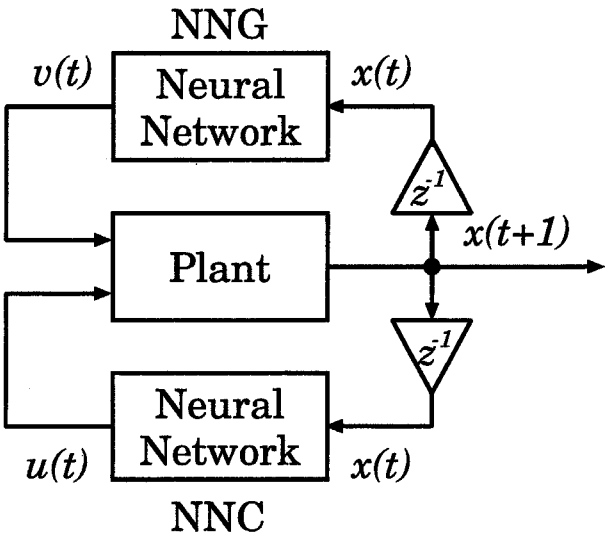


図 3.2: Block diagram for training robust controllers against ustructured uncertainties

あっても，本節の方法を適用することができる．

制御対象 (3.1) において，評価出力$\boldsymbol{z}(t)$ を次式のように定める．

$$\boldsymbol{z}(t) = \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad (\text{ただし} \boldsymbol{h}(\boldsymbol{0}, \boldsymbol{0}) = \boldsymbol{0}) \tag{3.6}$$

制御対象に不確かさが存在する場合，最も安全かつ信頼性の高いロバスト制御系の設計方針は制御系に対して最悪となる不確さを考慮し設計を行なうことである．しかし，最悪な不確さ$\boldsymbol{v}_{worst}(t)$ は制御系により異なるので，設計を行う際に既知とすることはできない．このために，前章で示したニューラルネットワークの学習によるフィードバック制御系の構築を (3.1) のようなシステムに適用することができない．不確かさ$\boldsymbol{v}(t)$ は外乱や制御対象のモデル化されていない部分であるが，(3.1) における最悪な$\boldsymbol{v}(t)$ は最適制御理論より制御対象の状態フィードバックの形で表すことができる．よって，制御系の設計の際に考慮すべき不確かさは状態の関数としてよい．そこで，不確かさを含む制御対象に対してロバストな制御系をニューラルネットワークを用いて学習させるために，図 3.2のように異なる二つのニューラルネットワークを用いる．一方は制御器として用いられるニューラルネットワークであり，$\boldsymbol{u}(t)$ を出力する．これに対して，もう一方は$\boldsymbol{v}(t)$ を出力し，制御対象の不確かさを模擬するために用いられている．本研究では (3.5) のような状態フィードバック制御器を学習することが目標である．また，不確かさ$\boldsymbol{v}(t)$ は状態変数$\boldsymbol{x}(t)$ の関数としてよいので，図 3.2に含まれる二つのニューラルネットワークにはいずれも階層構造型を用いることができる．

3.3.2 ロバスト制御系の学習

図3.2で表されるシステムにおける二つのニューラルネットワークの学習に，次のような評価関数 Jを用いることを考える．

$$J = \sum_{t=0}^{T} ||\boldsymbol{z}(t)||^2 - \gamma^2 ||\boldsymbol{v}(t)||^2 \tag{3.7}$$

ここで，$||\cdot||$ は信号のユークリッドノルムを表す．(3.7) に含まれているγは非負の定数であり，設計パラメータである．ここで，$\gamma = 0$ とすると (3.7) は最適制御問題の評価関数になることから，(3.7) を用いた学習は前章で示した最適フィードバック制御系の学習の拡張となっている．(3.7) を用いてロバスト制御系を学習させる場合には，$\gamma > 0$ のように選ぶ必要がある．

制御器として用いるニューラルネットワークは不確かさを含んだ制御対象を安定に制御できることを目的として，(3.7) を最小にするように学習を進める．これに対して，不確かさを模擬しているニューラルネットワークは制御器を含んだシステムにとって最悪の不確かさになるように，すなわち (3.7) 最大にするように学習を行う．その結果として，ロバスト制御系の学習は次の最適化問題に帰着される．

$$\min_{\boldsymbol{u}(t)} \max_{\boldsymbol{v}(t)} J \tag{3.8}$$

(3.7) に含まれるγは$\boldsymbol{v}$から$\boldsymbol{z}$への L_2ゲインに相当するので，スモールゲイン定理より$\boldsymbol{v}$のノルムが $1/\gamma$を越えない範囲までロバスト安定になる [30, 40]．よって，本手法ではγによって学習後のネットワークのロバスト性が定量的に保証され，小さなγを用いて学習を行なうことは制御系に大きなロバスト性を要求することを意味する．しかし，ロバストな制御系が学習可能なγには明らかに最小値があり，その最小値より小さなγで学習を行っても，ロバストな制御系を学習することはできない．γの最小値を用いて学習を行うと得られる制御系はロバスト性が最大になるが，一般にロバスト性が大きくなるに伴って制御性能が劣化する．このことを制御系が保守的になるという．不必要に大きなロバスト性を制御系に持たせるようとすると制御性能があまりよくない制御系が得られるので，本手法ではγを慎重に選ぶ必要がある．

(3.8) の最適化を行うことは難しいが，一方のニューラルネットワークを固定すると第2章において示した最適フィードバック制御器を学習と等価となる．そこで，(3.8) の最適化は一方のネットワークは固定し，他方のネットワークを前章に示した方法により学習を行った後に，固定したネットワークを学習させるというように，二つのニューラルネットワークの学習を交互に行う．(3.8) の変数は制御器としてのネットワークと不確かさを模擬するネットワークの結合度であるが，これは明らかに二つに分類することができる．このために (3.8) の最適化は前章で提案した学習アルゴリズムを応用することで十分に実現することができる．

制御器にとって最悪な不確かさが制御対象に存在すると仮定して設計を行なう方法は，もっとも安全な設計方法であり，H_∞制御 [30] などでも使われている考えである．しかし，ある制御器に対して最悪な不確

かさが実際には存在し得ないものとなることがある．実際に存在しない不確かさまで考えてしまうことは，制御性能の劣化など保守性という問題を引き起こす原因となる．本節で示した方法のように，不確かさのノルムのみを考慮する場合には，同一のノルムである不確かさが数多く存在することを配慮する必要がある．さらに，不確かさのノルムが大きくなるにつれて，不確かさの種類も多くなっていくため，γを小さくすることにより存在し得ない不確かさまで考慮してしまう可能性が大きくなる．その結果として，ロバスト性が大きな制御系を設計する場合には保守性が問題となる可能性が大きくなる．しかし，本節で述べた学習方法において保守性を軽減するには，学習の際に不必要に小さなγを用いないようにする以外に方法がない．しかし，適当な大きさのγを学習に用いたとしても，本手法では存在し得ない不確かさが考慮されることがあるため，得られる制御系がやはり保守的となる場合がある．これは，ノルムを用いて非構造的な不確かさを表す方法に一般に存在する欠点である．この欠点を解消するには考慮する不確かさに制限を加えることが必要となる．

3.3.3 定量的なロバスト性の解析

ロバストな制御系を構築するだけでなく，すでにある制御系のロバスト性を定量的に解析することもやはり重要である．ニューラルネットワークをはじめとして，インテリジェントシステムを応用した制御系では，そのロバスト性を定量的に解析する方法がなかったために，実際に利用する際に問題となることある．本節では，ニューラルネットワークを用いて，任意の制御系のロバスト性を定量的に解析する方法について述べる．

3.3.2節で述べた方法を少し変更することにより，ニューラルネットワークの学習を用いて任意の制御器のロバスト性の定量的な解析を行うことが可能となる．すでに述べたように，ロバスト制御系の学習では(3.8)の最適化を行なうことで学習を行うのに対して，制御器を固定し，(3.9)のように不確かさを模擬するネットワークのみ学習を行うことにより，制御系に対して最悪な不確かさを得ることができる．

$$\max_{\boldsymbol{v}(t)} J \tag{3.9}$$

ニューラルネットワークにより模擬された最悪の不確かさが存在しても制御系が安定であるなら，(3.9)の学習において用いたγをにより，制御系のロバスト性は少なくとも$1/\gamma$以上であることが分かる．この場合，制御系のロバスト性は余裕を持って評価されており，安全な評価方法であるといえる．また，最悪の不確かさの存在により制御系が不安定となるまで繰り返して学習を行いγを減少させていくことにより，γの最小値を求めることができる．γの最小値が制御系のロバスト性の定量的な指標になる．現在のところ制御系が安定であるγの最小値を求める効率のよい方法は存在せず，二分法などの繰り返し計算を行なう必要がある．しかし，実際にはγの厳密な最小値が必要となることはほとんどないと考えられ，安全側に余裕を持っ

た評価で十分であると思われる．

本手法でロバスト性の定量化できる制御器はニューラルネットワークで実現されているものだけでなく任意のものでよい．第2章における最適フィードバック制御系を学習したニューラルネットワークのロバスト性はこれまで測定することはできなかったが，本方法により初めて可能となる．

3.3.4 計算例

ここでは3.3節において説明した手法により，学習を行なった例を示す．制御対象の方程式を以下の通りとする．ここで，h は0.01とする．

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} x_1(t) + hx_2(t) \\ x_2(t) + h\left\{\left(1 - x_1^2(t)\right)x_2(t) - x_1(t) + u(t) + v(t)\right\} \end{bmatrix} \tag{3.10}$$

$$z(t) = [x_1(t), x_2(t), u(t)]^T \tag{3.11}$$

また，初期条件を $(x_1(0), x_2(0)) = (0.5, 0)$ として学習を行なった．学習の際に用いる評価関数は次の通りである．

$$J = \sum_{t=0}^{1000} \left[||z(t)||^2 - \gamma^2 v^2(t)\right] \tag{3.12}$$

学習には，それぞれ中間ユニット数が10個であるニューラルネットワークを用いた．本手法により学習を行なったニューラルネットワークと比較を行うために，制御対象 (3.10) において $v(t) \equiv 0$ と仮定し，評価関数 (3.12) において$\gamma = 0$ として，最適フィードバック制御系の学習を行った．3.3節で説明したロバスト性を定量的に測定する方法を最適フィードバック制御系を学習したニューラルネットワークに対して適用したところ，$\gamma = 2.832$ を得た．これに対して，(3.12) において$\gamma = 1.549$ としてロバスト制御系の学習を行った．ここで用いたγの値はロバスト制御系を学習することが可能な最小値であった．つまり，学習により得られた制御系のロバスト性は最大となっている．γを比べることにより，ロバスト制御系を学習したニューラルネットワークは最適フィードバック制御系を学習したものに比べて，およそ2倍のロバスト性を持っていることが分かる．不確かさ $v(t)$ が存在しないと仮定し，制御対象を学習済の二つのニューラルネットワークを用いて制御した結果を3.3に示す．この図を見れば分かるように，ロバスト制御系を学習したニューラルネットワークは最適フィードバック制御系を学習したものよりも応答が遅くなっており，制御性能が劣っていることが分かる．これは大きなロバスト性を確保したために，公称対象における制御性能が劣化したためである．学習にもう少し大きなγを用いれば，このような制御性能の劣化は軽減される次に，制御対象に大きな不確かさが存在した場合の制御結果を図3.4に示す．ここでは，不確かさ $v(t)$ により制御対象 (3.10) の $x_2(t)$ に関する状態方程式が次のように変化したとした．

$$x_2(t+1) = x_2(t) + h\left\{1.5(1 - x_1^2(t))x_2(t) - 1.5x_1(t) + 0.5u(t)\right\} \tag{3.13}$$

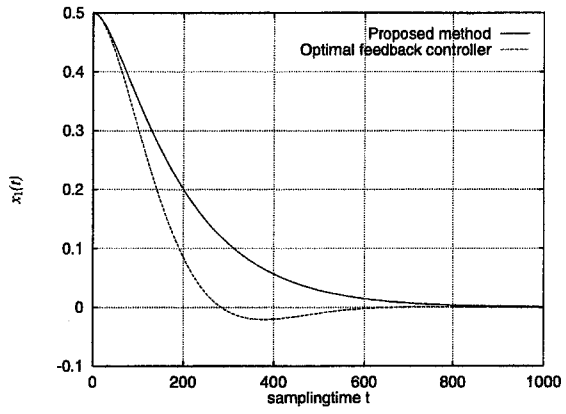


図 3.3: Responses of the nominal plant

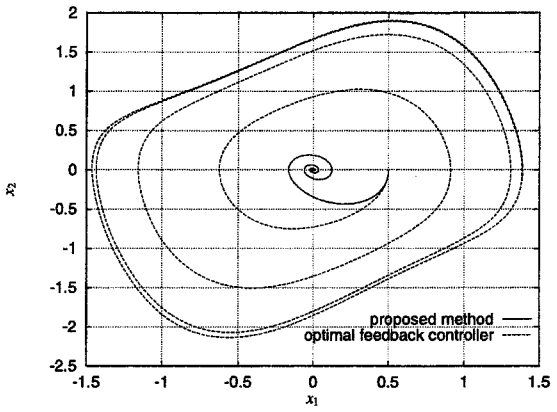


図 3.4: Responses of the plant with large uncertainty

図 3.4から分かるように，不確かさの存在のために最適フィードバック制御系は制御対象を安定に制御することができないが，本手法により学習を行ったニューラルネットワークによると，不確かさが存在している場合であっても安定に制御ができていることが分かる．また，γによって，提案した手法により学習したネットワークのロバスト性が高いことが示されていたが，この結果はそれを裏付けるものである．

3.4 構造的な不確かさに対するロバストな制御系

3.4.1 ロバスト制御系の学習

不確かさの構造的な情報が得られているならば，制御系は構造的な情報により制約された不確かさの存在範囲に対してロバスト性を有すればよいことになる．このために，不必要に大きなロバスト性を制御系に要求することがなくなるので，その結果として保守性を軽減することが可能となる．しかし，3.3節で説明した方法では不確かさの大きさのみを考慮し，構造的な情報は有効に利用できない．不確かさの構造的な情報が存在する場合に，非構造的な不確かさに対するロバスト制御系の学習法を用いるには，不確かさに関する構造的な情報を不確かさのノルムの上限という情報に変換する必要がある．しかし，その際に不確かさに関する構造的な情報は失われてしまい，その結果として不確かさの存在範囲は大きくなってしまう．このために，学習において実際に存在しないために考慮しなくてもよい不確かさまでが考慮され，学習後のニューラルネットワークが保守的な制御性能となりがちになる．よって，先験的に得られている不確かさの構造的な情報を有効に活用することが可能な学習方法が必要である．構造的な情報をもつ不確かさはパラメータの不確かさに帰着することができるので，制御対象は (3.2) あるいは (3.3) のように表すことが可能である．パラメータの不確かさを考慮するロバストな制御系の学習法は，保守性の低減のための有効な

手法となると考えられる．

不確かなパラメータ$\boldsymbol{\alpha}$は時間的に変動する場合と変動しない場合の二つに大別することが可能である．不確かなパラメータが時間的に変動することを考慮して設計を行なった制御系は，不確かなパラメータは時不変であるとして設計を行なった制御系に比べて，一般的にロバスト性が高くなることは明らかである．しかし，不確かなパラメータは時間変動しない，あるいは変動する場合であってもその変化速度が制御系の応答に比べて十分に遅いときに，制御系の応答速度と同程度の変動速度を持つパラメータの変化を考えることは冗長であるために，学習により得られた制御系が保守的になる恐れがある．そこで，不確かなパラメータが時変である場合と時不変である場合のそれぞれに対して，ニューラルネットワークの学習によりロバストな制御系の構築を行なう方法を導く必要がある．実際にパラメータの時間変動を考慮する場合であっても，制御系に対して最悪なパラメータは時間変化しない一定値となる可能性もある．このために，時不変な不確かなパラメータを考慮する学習法は時変な不確かなパラメータを考慮する学習法の特別な場合となる．

3.4.2 不確かなパラメータが時変である場合

不確かなパラメータ$\boldsymbol{\alpha}$が時間的に変動する場合，制御器にとって最悪な変動パラメータの時系列が存在するが，それは (3.14) で表されるような評価関数 L を最大にする$\boldsymbol{\alpha}$である．

$$L = \sum_{t=0}^{T} L_0(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{3.14}$$

ここで，(3.14) における L_0は正定関数である．制御器をある状態フィードバック制御器 ($\boldsymbol{u}$=$\boldsymbol{u}(\boldsymbol{x})$) と固定すると，

$$\max_{\boldsymbol{\alpha}(t)} L(\boldsymbol{x}(t), \boldsymbol{u}(\boldsymbol{x}(t))) \tag{3.15}$$

$$\textit{subject to} \quad \begin{cases} \boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(\boldsymbol{x}(t)), \boldsymbol{\alpha}(t)) \\ \boldsymbol{\alpha}(t) \in \Delta \end{cases} \tag{3.16}$$

という最適制御問題の解が最悪パラメータ$\boldsymbol{\alpha}(t)$ となる．特に，制御対象の状態方程式が (3.3) のように不確かなパラメータに関して線形であるとき，最悪なパラメータ$\boldsymbol{\alpha}$は存在範囲Δのある頂点から別の頂点へとBang-Bang 制御のようにある時刻を境にして急激に切り換わるような変化をする．制御系設計の際に，制御器にとって最悪なパラメータ変動が生じると想定し，そのときの性能を向上するように制御系を設計することにより，パラメータ変動に対してロバストであり，もっとも安全な制御系が構築できる．よって，制御系設計問題は次の Mini-Max 最適化問題に帰着される．

$$\min_{\boldsymbol{u}(t)} \max_{\boldsymbol{\alpha}(t)} L \tag{3.17}$$

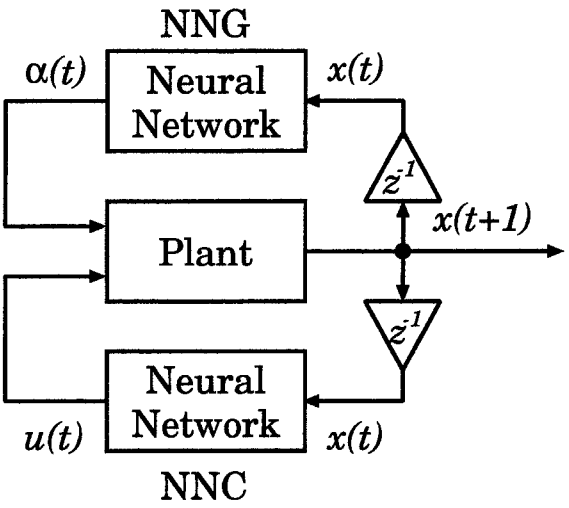


図 3.5: Block diagram for training robust controllers against time varying uncertain parameters

3.3節と同様に図3.5のように異なるニューラルネットワークを二つ用い，それぞれを競合するように学習を行なうことによりパラメータ変動に対してロバストな制御系が学習できる．図3.5において，一つのネットワークはフィードバック制御器であり，他方は制御系に対する最悪なパラメータ変動を模擬するものである．先に述べたように，最悪なパラメータはフィードバック制御器を固定すると，最適制御問題 (3.15)(3.16) を解くことにより求められるが，Tを十分に大きいとすると$\boldsymbol{\alpha}(t)$ は$\boldsymbol{x}(t)$ の関数としてよいので，不確かなパラメータを出力するネットワークへの入力は$\boldsymbol{x}(t)$ とすることが適当である．実際に，3.4.5節で挙げる数値例において，ネットワークへ$\boldsymbol{x}(t)$ 以外に時刻 t も陽に入力することも試みたが，結果に差はほとんど見られなかった．

また，制御器を固定して，最悪パラメータを出力するニューラルネットワークのみを学習させることにより，制御系を学習したニューラルネットワークに対する最悪なパラメータ時系列を求めることが可能である．よって，ニューラルネットワークの学習により，任意の制御系が構造的な変動に対してロバスト性を有するかどうかを検証することもできる．

3.4.3 不確かなパラメータが時不変の場合

不確かなパラメータが時間変動しないことが分かっている場合に，3.4.2節で説明した手法により制御系の学習を行うと，時間変動するパラメータの不確かさまで考慮することになるため，過度のロバスト性が確保される可能性がある．その結果として，制御性能が保守的となってしまう可能性があるので，時間変動をしない不確かなパラメータを考慮した制御系が必要である．そこで，本節では不確かな時不変パラメー

タを持つ制御対象に対し，ロバスト制御系をニューラルネットワークの学習により構築する手法について説明をする．

ある制御系により制御を行う際に，不確かなパラメータ$\boldsymbol{\alpha}$によって制御結果は異なり，Δ内に評価関数(3.14)により定量化された制御性能を最悪にするパラメータ$\boldsymbol{\alpha}_{worst}$が存在する．制御器にとって最悪なパラメータ変動が生じると仮定し，そのときの性能を向上するように制御系を設計する方法が最も信頼性の高いロバスト制御系の設計の指針であるので，本節でもこの指針を適用する．すなわち，最悪パラメータ下にある制御対象を最良に制御するように制御系を学習することにより，パラメータの不確かさに対してロバストな制御系が構築できる．3.3節や3.4.2節と異なり本手法において用いるニューラルネットワークは一つであり，学習も図3.1で示されるブロック図にて行なう．学習に用いる評価関数を(3.18)ように設定することにより，時不変なパラメータの不確かさに対してロバストな制御系を学習することが可能となる．

$$J = \max_{\boldsymbol{\alpha}\in\Delta} L \tag{3.18}$$

特に，制御対象の状態方程式が(3.3)のように不確かなパラメータに関して線形であるとき，最悪なパラメータ$\boldsymbol{\alpha}$は一般にΔの頂点$V(\Delta)$となるので

$$J = \max_{\boldsymbol{\alpha}\in V(\Delta)} L \tag{3.19}$$

とすることにより，学習に必要な計算の省略化を図ることができる．

3.4.4 学習アルゴリズム

本節では，構造的な不確かさに対してロバストな制御系の学習のための学習アルゴリズムについて考察する．3.4.3節において提案した学習方法において用いる評価関数(3.18)は明らかに微分不可能であるので，勾配法による学習アルゴリズムを用いることはできない．このために前章で示したPowellの共役方向法[26]に基づく学習アルゴリズムを採用する必要がある．また，3.4.2節において示した学習方法では，用いる評価関数(3.14)に現われるL_0および対象の状態方程式がそれぞれ各変数に対して少なくとも一階微分可能であれば，勾配を計算することが可能となり，勾配法に基づく学習アルゴリズムを適用することが可能であるように思われる．しかし，前章で示したように勾配法による学習アルゴリズムでは任意のBang-Bang解の勾配が0となるので，最短時間制御問題に対して勾配法に基づく学習アルゴリズムを用いると，学習が局所解に陥ったり，解を発見できる場合でも学習速度が遅く，長い計算時間が必要となるなどの欠点を持っていることが分かっている．ここで，評価関数(3.14)を最悪にするようなパラメータ変動はBang-Bang制御のように急激な切り換えを伴うものなることが多いことから，3.4.2節の学習方法も最短時間制御問題と同様に，勾配法に基づく学習アルゴリズムが向かないと考えられる．よって，時変なパラメータの不確か

さを伴う制御系に対するロバスト制御系の学習においてもPowellの共役方向法に基づく学習アルゴリズムを採用するほうが効率がよい．また，制御対象のヤコビアンを求めることが不要となることは学習を行うため労力の短縮に大きく貢献する．このような優位性のために，後に示す数値シミュレーションでは全てPowellの共役方向法に基づくアルゴリズムを用いて学習を行なった．

3.4.5 数値例 1

提案した手法を適用した数値例を示す．制御対象は(3.20)とする．ただし，(3.20)に含まれるc, k, bはパラメータであり，これらの公称値は全て1であるとする．

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} x_1(t) + 0.01x_2(t) \\ x_2(t) + 0.01\left\{c\left(1.0 - x_1^2(t)\right)x_2(t) - kx_1(t) + bu(t)\right\} \end{bmatrix} \tag{3.20}$$

変動する不確かなパラメータの場合

まず，(3.20)に含まれるc, k, bは不確かで，時間とともに変動すると仮定する．ここで，各パラメータの変動幅は公称値の±20%であるとする．まず，公称パラメータである制御対象に対して次のような評価関数，初期条件でニューラルネットワークを用いて最適フィードバック制御系の学習を行った．

$$L = \sum_{t=0}^{1000} x_1^2(t) + x_2^2(t) + u^2(t) \tag{3.21}$$

$$(x_1(0), x_2(0)) = (0.5, 0) \tag{3.22}$$

ニューラルネットワークを用いた最適フィードバック制御系にとって，最悪なc, k, bの変動パターンを3.4.2節で提案した手法を用いてニューラルネットワークに学習させた．図3.6にその結果を示すが，3.4.2節で述べたように最悪なパラメータ変動パターンはBang-Bang解のように境界値から境界値へ切り換わるものであった．

次に3.4.2節で提案した方法を用いて，パラメータ変動に対してロバストな制御系をニューラルネットワークに学習させた．このとき，ロバスト制御系を学習したニューラルネットワークに対する最悪なパラメータ変動は一定の境界値($c = k = 1.2,\ b = 0.8$)であった．パラメータ変動を考慮した学習を行なったネットワークによる制御結果と公称制御対象に対して最適フィードバック制御系を学習したネットワークによる制御結果を図3.7，3.8に示す．図より，提案した手法で学習したロバスト制御系のほうがパラメータの変動に対して頑強であることが分かる．また，ここで示したロバスト制御系の学習例では最悪な不確かなパラメータは時間的に変化せずにある境界値となったが，初めから不確定パラメータは時間的に変化しないと仮定し3.4.3節の方法を用いると，パラメータが時間的に変動することに対するロバスト性は保証されないので，注意が必要である．

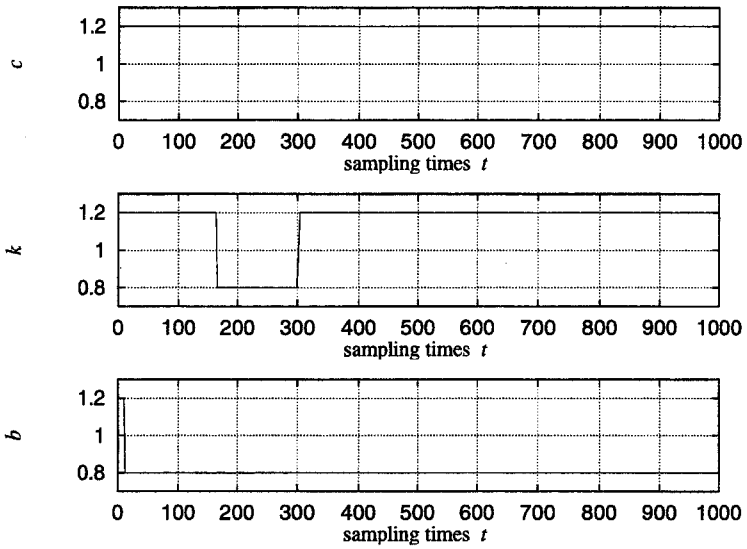


図 3.6: The worst time series of uncertain parameters for the neural network trained to be an optimal feedback controller

時間的に変化しない不確かなパラメータの場合

本節では，(3.20) に含まれる c, k, b はすべて時間的に変化しないとする．ここでは，学習のときに仮定する不確かなパラメータの存在範囲により，学習後のネットワークのロバスト性がどのように変化するかを詳しく調べる．学習に用いる評価関数 (3.18)，または (3.19) に含まれる L として，a を正の定数パラメータを含んだ (3.23) を用いる．また，制御入力は (3.24) のように，その使用に制限があるとする．

$$L = \sum_{t=0}^{T} a x_1^2(t) + x_2^2(t) \tag{3.23}$$

$$|u(t)| \leq 1.0 \quad (^\forall t \geq 0) \tag{3.24}$$

状態方程式 (3.20) で表される制御対象に対して (3.23) のような評価関数の形を考えると，その最適解は特異解を含んだものとなる [22,33]．3.6節で述べるように，最適特異解では状態空間中のある多様体の上に状態が拘束される，いわゆるスライディングモード制御となる．スライディングモード制御に関しては [35–37] において詳細に説明されている．

3.4.3節で提案した学習方法において，パラメータの変動範囲は定量的に仮定され，不確かなパラメータが仮定された範囲内である限りロバスト安定である制御器を学習する方法である．その結果，得られる制御器はそのロバスト性は定量的に保証されていることになるが，さらに一般的にそのロバスト性を定量的に解析することを試みる．3.3節で提案した方法を用いると，任意の制御系の一般的なロバスト性が定量的

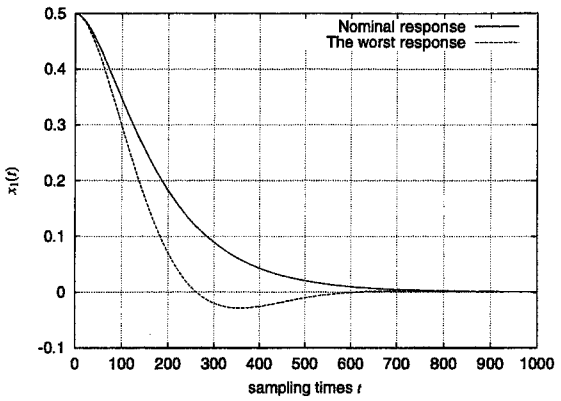


図 3.7: Nominal response and the worst response of a robust control system

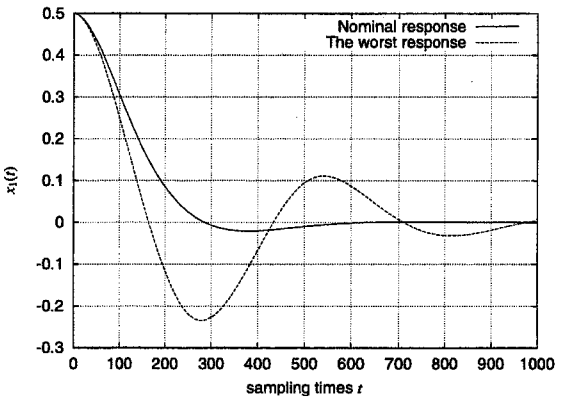


図 3.8: Nominal response and the worst response of an optimal control system

に評価できる.

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} x_1(t) + 0.01x_2(t) \\ x_2(t) + 0.01\left\{\left(1.0 - x_1^2(t)\right)x_2(t) - x_1(t) + u(t) + v(t)\right\} \end{bmatrix} \tag{3.25}$$

ここでは, (3.25) のようにパラメータの不確かさだけでは表現できない非構造的な不確かさ $v(t)$ が制御対象の公称モデルに含まれるとして, $v(t)$ がどれぐらいの大きさまで制御系がロバスト安定かどうかを調べる. 3.3節の方法では, 二つのニューラルネットワークを競合させるように学習させるが, このための評価関数は

$$J = \sum_{t=0}^{T} ax_1^2(t) + x_2^2(t) - \gamma^2 v^2(t) \tag{3.26}$$

とする. 学習を行うシステムのブロック図は図 3.2と同じものである. ここでγは正の定数パラメータであり, 不確かさ vから規範出力までの L_2ゲインに相当し, 小さいほど制御対象のロバスト性が高いことを示す. よって, γをある値に定めて学習させた結果, 制御器がそのγにおける最悪の不確かさを模擬したニューラルネットワークを含んだ対象を安定化できれば, 制御器のロバスト性は少なくともγであることがいえる. また, 明らかにロバスト安定であるγの大きさには最小値が存在するので, 繰り返し計算によりそれを求めることにより提案手法で学習したニューラルネットワークのロバスト性の限界を定量的に評価することができる.

ここでは, b には不確かさが含まれず, 不確かなパラメータは c, kの二つであるとして学習を行なったものと, c, k, b の全てが不確かであると仮定して学習を行なったものとの比較を行なう. また, 学習の際に不確かさによる変動範囲を 10%, 20%, 30%の三種類を仮定して学習を行った. また, (3.23) に含まれる a は 1, 2 の二種類として学習を行なった. それぞれのニューラルネットワークのロバスト性を図 3.9に示す. 明らかなように, 不確かなパラメータの存在領域を大きく仮定すると学習後のネットワークのロバスト性は

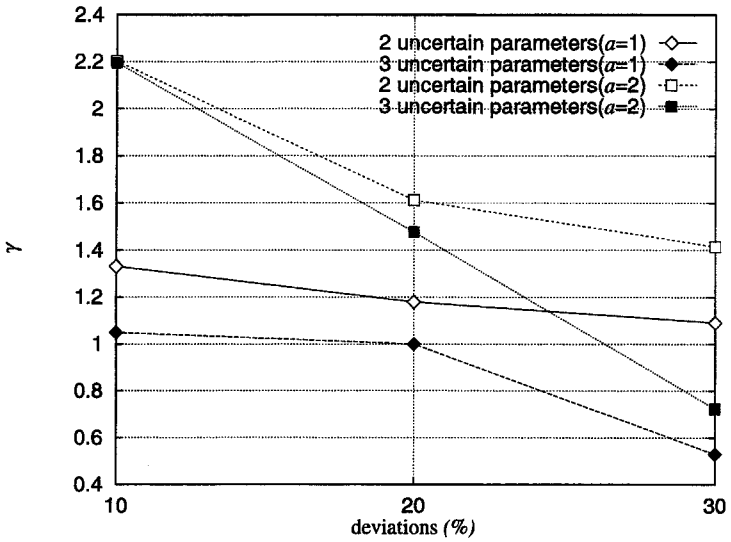


図 3.9: Robustness of the trained neural networks

高くなり，設計要求に応じたロバスト性を実現していることが分かる．また，評価関数 (3.23) に含まれる a は学習後のネットワークの制御性能を表す設計パラメータであり，大きいほど状態の収束が早くなるために制御性能が高くなるが，a によってロバスト性が変化することも図 3.9 より明らかであり，大きな a を用いて学習を行なうことにより制御性能を向上させると，それに応じてロバスト性が低下してしまうことが分かる．

3.4.6 数値例 2

制御対象の方程式を以下の通りとする．ここでは h は 0.01 とした．

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ x_3(t+1) \\ x_4(t+1) \end{bmatrix} = \begin{bmatrix} x_1(t) + h \cdot x_2(t) \\ x_2(t) + h(-G \cdot \sin x_3(t) + x_1 x_4(t)^2) \\ x_3(t) + h \cdot x_4(t) \\ x_4(t) + h \cdot I \cdot u(t) \end{bmatrix} \tag{3.27}$$

$$|u| \leq 1.0, \quad \boldsymbol{x}(0) = (2,0,0,0) \tag{3.28}$$

この制御対象には不確定パラメータ G, Iが存在している．G は公称値が 9.8 で±10%の偏差を，Iは公称値が 1.0 で±20%の偏差を含んでいるとする．

$$G \in [8.82, 10.78] \quad , \quad I \in [0.8, 1.2] \tag{3.29}$$

学習に用いる評価関数 (3.18)，あるいは (3.19) に含まれる J_0は次のように定めた．

$$J_0 = \frac{1}{2}\sum_{t=0}^{2000}\left\{100x_1(t)^2 + x_2(t)^2 + x_3(t)^2 + x_4(t)^2 + u^2(t)\right\} \tag{3.30}$$

(3.30) のように評価関数を定めると応答が早い制御器を要求するが，一般にロバスト性と制御性能は相反する要求である．このために，公称制御対象を用いて (3.30) により最適フィードバック制御系を学習したニューラルネットワークでは，パラメータの変動により不安定になりやすい．3.3.4節の例と同様に，公称パラメータに対して (3.30) を評価関数として最適フィードバック制御系を学習させたニューラルネットワークと 3.4.3節において示した方法により学習を行なったネットワークの比較を行なった．予想されたように最適フィードバック制御系の学習を行ったニューラルネットワークはパラメータ変動に対して非常に弱くなっていた．パラメータが変動したときの，最適フィードバック制御系を学習したニューラルネットワークによる評価関数の変化を図 3.10に示す．また，その等高線を図 3.11に示す．ただし，ここは評価関数の $\log_{10}$をとったものを示した．これにより，最適フィードバック制御系を学習したニューラルネットワークはわずか 1%のパラメータ変動すらも許容できず，安定に制御を行うことができなかったことが分かる．しかし，パラメータの変動を考慮してロバスト制御系の学習を行ったニューラルネットワークでは，(3.29) で表された領域でパラメータが変動しても安定に制御をすることができた．図 3.12に公称パラメータの場合の応答を示す．この図を見れば分かるように，公称パラメータの場合であっても，本手法で学習したニューラルネットワークは最適フィードバック制御系を学習したネットワークと比べてそれほど制御性能が劣化せず良好な制御結果を示すことが分かる．3.3.4節の例ではロバスト性の向上に伴って制御性能の劣化が見られたが，不確かさの構造的な情報を利用することにより，制御性能に優れたロバスト制御系が学習できることが分かる．図 3.13にロバスト制御系を学習したニューラルネットワークにとって最悪なパラメータの場合の応答を示した．最適フィードバック制御を学習したネットワークではまったく制御できていないが，ロバスト制御系を学習したニューラルネットワークは最悪のパラメータが生じても安定に制御できている．パラメータが変動したときの，ロバスト制御を学習したニューラルネットワークによる評価関数の変化を図 3.14に示す．また，その等高線を図 3.15に示す．ただし，ここでも評価関数の $\log_{10}$をとったものを示した．図 3.14の中央が公称パラメータであるが，(3.30) で示されたパラメータの存在範囲全体において評価関数があまり大きく変化していないことが分かる．以上により，3.4.3節において示した学習法により，パラメータの変動による評価関数の変動を小さく抑えることができる制御系が得られていることが分かる．

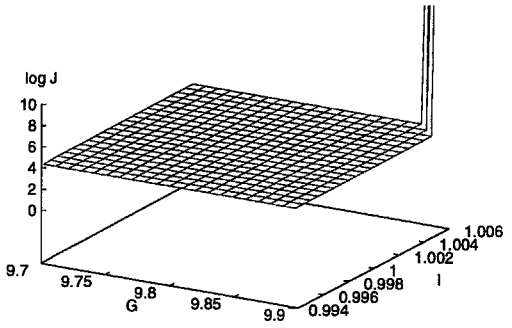


図 3.10: Performance index in a small region(optimal control)

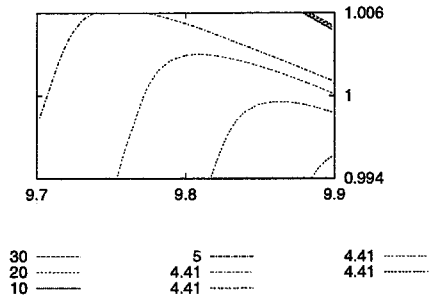


図 3.11: Performance index in a small region(optimal control)

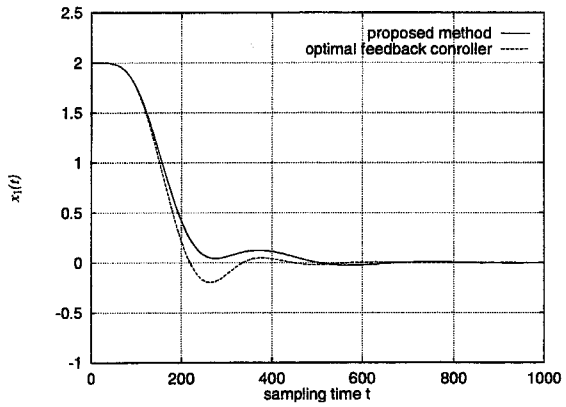


図 3.12: Responses of the nominal plant

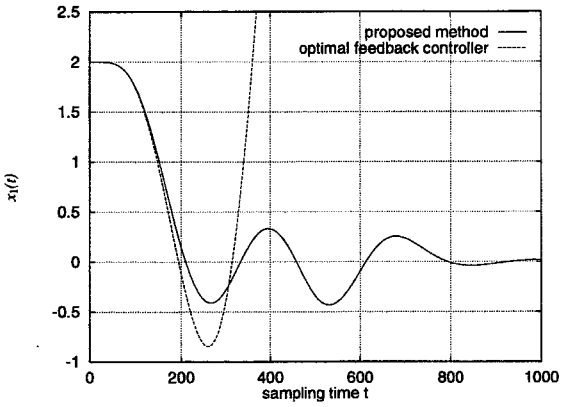


図 3.13: Responses under the worst parameters

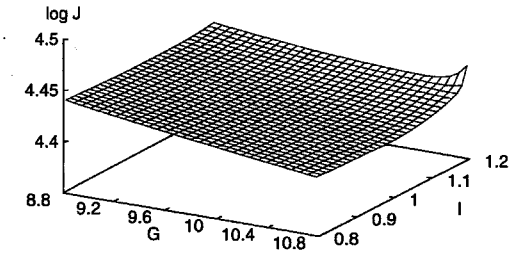


図 3.14: Performance index in Δ(proposed method)

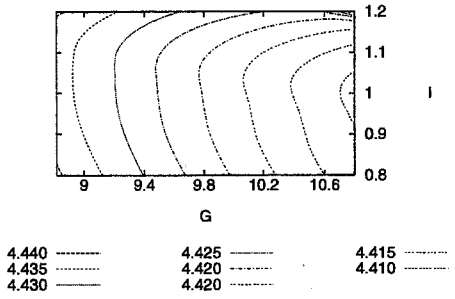


図 3.15: Contour of the performance index in Δ(proposed method)

3.5 学習に用いる初期状態の影響

ニューラルネットワークを用いて最適フィードバック制御系，あるいはロバスト制御系を学習・構築する際には，制御対象の初期状態を定めて学習を行なうことが多い．この場合，他の初期状態から始まる場合には，ニューラルネットワークの汎化能力に期待することになる．前章で述べたように最適フィードバック制御系を学習したニューラルネットワークは良好な汎化性を持っていることが確認されている．しかし，評価関数の差という評価をしか行なっていないために，制御系として本当に望ましい汎化能力を持っているのかどうかは不明である．また，[33] により最適フィードバック制御系を学習したニューラルネットワークのロバスト性が期待される値より低くなっていることがあることを指摘した．この結果は，制御系として用いるニューラルネットワークの汎化能力として，学習を行った状態と異なるときの制御性能を考えているだけでは不十分であることを意味する．なぜならば，学習後のニューラルネットワークが適切な汎化能力をもっているならば，ロバスト性を全く考慮しない学習方法であっても，期待通りのロバスト性をニューラルネットワークが持つはずだからである．このようにロバスト性と汎化能力は密接な関係を持っている．3.3節で述べたロバスト性の解析方法では，制御器の最大の弱点となる不確かさが別のニューラルネットワークによって学習されることにより，ロバスト性が定量的に評価された．この方法によりロバスト性の定量評価を行うと，ニューラルネットワークが有している汎化能力の最大の弱点が別のニューラルネットワークの学習によって発見される．よって，制御器として用いるニューラルネットワークの汎化能力の定量的な指標として，制御性能だけでなく，ロバスト性を加えることが適切であると考えられる．

ロバスト制御系の学習の場合には，不確かさが学習の際に経験する解軌道を多様とするような効果を持つために，単一の初期状態を用いることの影響は小さくなるが，やはりその影響を無視することはできない．制御対象の不確かさとは少し問題が異なるが，制御対象の初期状態の不確かさもやはり考慮しなければならない．このような制御対象の初期状態の不確かさはパラメータの不確かさとして取り扱うことが可能である．学習に用いる初期状態$\boldsymbol{x}_0$の存在する既知の閉領域をX_0とすると，評価関数 (3.14) により定量化された制御性能は一般にX_0の境界∂X_0に含まれる初期状態で最悪となるので，次のように評価関数を定めてニューラルネットワークを学習させることで，最悪性能の改善を図ることができる．

$$J = \max_{\boldsymbol{\alpha} \in \partial X_0} L \tag{3.31}$$

このように制御対象の初期状態の不確かさを考慮した学習を行なうことにより，学習に用いる初期状態に対する曖昧さを解決することができる上に，学習後のネットワークの制御性能劣化の防止，あるいはロバスト性減少の防止が期待できる．

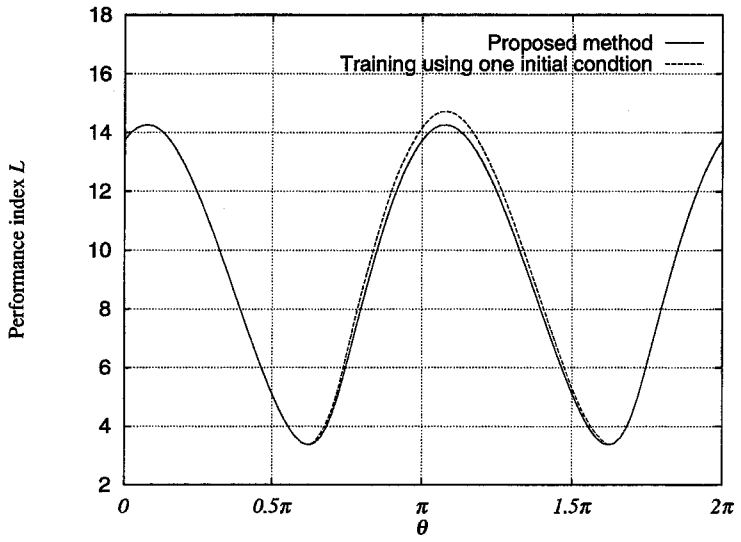


図 3.16: Effect of the initial condition

3.5.1 数値例

ここでは，学習に用いる初期条件がロバスト性や制御性能に与える影響を数値計算によって調べる．制御対象は (3.20) であるとする．ただし，(3.20) に含まれる c, k, b はパラメータであり，これらの公称値は全て 1 であるとする．(3.20) に含まれる c, k, b は不確かで時間的に変化せず，変動範囲は公称値の±10%であるとする．また，制御対象の初期状態は，原点を中心にして半径が 0.5 である円板内にあると仮定し，3.4.3 節と 3.5節の方法を複合し評価関数 L を (3.32) としてニューラルネットワークの学習を行った．

$$L = \sum_{t=0}^{1000} x_1^2(t) + x_2^2(t) \tag{3.32}$$

3.3節において述べた方法により，このネットワークのロバスト性を定量的に評価したところ，$\gamma = 1.192$ を得た．これに対して，初期点を $(0.5, 0)$ と一点に定めて学習を行なったネットワークのロバスト性を測定すると $\gamma = 1.049$ を得た．γの値の比較から分かるように，単一の初期状態を用いて学習を行ったネットワークの方がロバスト性が高くなっていた．この原因を調べるために，初期状態を

$$(x_1(0), x_2(0)) = (0.5\sin\theta, 0.5\cos\theta) \quad \theta \in [0 : 2\pi] \tag{3.33}$$

と変化させたときの，公称パラメータである制御対象の応答より計算した評価関数 L を図 3.16に示す．この図によると，適当な初期点を仮定して学習を行なったネットワークはその初期点近くでは十分な制御性能を持っているが，初期点が遠ざかるにつれて制御性能がすこし劣化することが分かる．これは，学習のときに経験していない状態変数の領域を通るようになるためである．この数値例では，単一の初期状態を用い

て制御系の学習を行ったニューラルネットワークでは.，制御性能が劣化していたために，初期状態を不確かなパラメータとして学習をさせたニューラルネットワークよりもロバスト性が大きくなった．この結果が示すように，適当な単一の初期点から学習を行なったものは不確かさを考慮する学習を行なったとしても，そのロバスト性あるいは制御性能を損なう可能性がある．

3.6 スライディングモード制御系の学習

3.6.1 スライディングモード制御

近年，注目を集めているロバスト制御系の一つとしてスライディングモード制御 [35–37] があるが，これは状態空間中に設計した超平面 (滑り面) に状態を拘束させるように制御を行う非線形制御系である．スライディングモード制御では，ある切換面において入力値が不連続に変化するので，リレー制御の一種であるといえる．また，対象の状態によって制御器の切り換えを行なうことから，可変構造制御とも呼ばれる．スライディングモード制御はマッチング外乱の除去能力などから高度なロバスト性を有することが知られており，非線形ロバスト制御として最も有力なものの一つであると考えられている．しかし，従来の方法では制御入力の大きさの制限を考慮することができないために，滑り面に状態を拘束するためには大きな入力が必要となることが多く，入力の大きさに厳しい制限のある対象に対しては状態を滑り面に拘束できず，実際には用いることができないという難点があった．この難点を解消するために，[38] では非線形滑り面を設計している．

ここではスライディングモード制御の基本について簡単に述べる．スライディングモード制御では滑り面の方程式を$\sigma(\boldsymbol{x}) = 0$とすると，制御入力 u を

$$u = U_{equiv} - K\mathrm{sgn}(\sigma) \tag{3.34}$$

あるいは

$$u = -K\mathrm{sgn}(\sigma) \tag{3.35}$$

で与える．U_{equiv}は等価制御入力と呼ばれ，状態フィードバック制御則で実現される．(3.34)，(3.35) における Kは正の定数であり，sgn(·) は符号関数である．このようにスライディングモード制御器とは滑り面を入力切換面として，入力の切り換えを行うものである．しかし，実際には物理的な制限のために (3.34)，(3.35) のように，入力を急激に切り換えることはできない．また，入力の急激な切り換えはチャタリングを引き起こす原因となるので，多くの場合には切換面に境界層を設けることにより，その中で入力が連続的に変化するように入力の連続化を行う．もし，滑り面への到達則が満たされていれば，状態は滑り面へ有限時間で到達し，その後は状態は滑り面に拘束されながら原点へと遷移する．しかし，(3.34) のような制御則は，等価制御入力のために入力に飽和特性があるような対象において，その限界を越える制御入力を要求することが多い．一方，(3.35) のような制御則では滑り面や初期状態によっては状態を滑り面に拘束できない．よって，従来の方法では，入力に厳しい制限のある対象に対してスライディングモード制御を用いることは困難であるといえた．

本節では，3.6.2節で述べる最適制御問題の特異解とスライディングモード制御系の対応に注目し，ニューラルネットワークの学習によりスライディングモード制御系を構築する方法を述べ，これにより大きなロバスト性を持つ制御系を学習することができることを示す．また，本手法では従来の方法において困難であった入力の飽和特性を容易に取り扱うことが可能である．入力の飽和特性を考慮することにより，線形制御対象，かつ線形滑り面を設計に用いる場合であっても，入力切換面は状態空間中で曲がっていることを数値例により示す．さらに，従来法では難しかった非線形滑り面の設計とそれに対応する制御系の構築も容易となることを示す．

3.6.2 最適特異制御とスライディングモード制御系

本節において，取り扱う制御対象は (3.36) のように入力に関しては線形なモデルで表されるとする．

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} \tag{3.36}$$

ここで，$\boldsymbol{x}$はn次の状態変数ベクトル，$\boldsymbol{u}$はm次の入力変数ベクトル，$\boldsymbol{f}$はn次のベクトル関数，$\boldsymbol{g}$は$(n \times m)$行列である．また，入力$\boldsymbol{u}$は次のように飽和特性を持つと仮定する．

$$U_i^{\min} < u_i < U_i^{\max} \quad (i = 1, 2, \cdots, m) \tag{3.37}$$

ここで，添字iはベクトルのi成分を表す．このような制御系に対して状態フィードバック制御器を構成し，状態を原点に移動させることを目的とする．

(3.36) で表される制御対象に対して，次のように入力$\boldsymbol{u}$を陽に含まない評価関数Jを最小 (あるいは最大) にする最適制御問題を考える．

$$J = \int_0^{T_c} L(\boldsymbol{x})dt \tag{3.38}$$

ここで，T_cは十分に大きな正の数である．実際にニューラルネットワークの学習は離散時間系で行われるため評価関数 (3.38) は

$$J_d = \sum_{t=0}^{T_d} L(\boldsymbol{x}(t)) \tag{3.39}$$

のように変換されたものに基づいて行う．T_dはサンプリング時間幅をhとすると$T_d = T_c/h$である．この系の Hamiltonian Hはn次随伴変数ベクトルを$\boldsymbol{\lambda}(t)$として

$$H(\boldsymbol{x}, \boldsymbol{u}) = L(\boldsymbol{x}) + \boldsymbol{\lambda}(t)^T \left(\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u}\right) \tag{3.40}$$

$$= H_1(\boldsymbol{x}, \boldsymbol{\lambda}) + H_2(\boldsymbol{x}, \boldsymbol{\lambda})\boldsymbol{u} \tag{3.41}$$

となり，入力$\boldsymbol{u}$の一次式となる．最大値原理と入力の飽和特性 (3.37) より，最適な制御入力は最大入力あるいは最小入力となることが考えられる．すなわち，最適解はいわゆる Bang-Bang 解となると考えられる．

しかし，(3.40) において$\boldsymbol{u}$の係数に相当する部分がある時間区間で 0 となる場合は，$\boldsymbol{u}$を Hamiltonian が最小 (あるいは最大) という条件から決定することが不可能である．このような状態を特異であるといい，このときの最適解を最適特異解 [39] という．最適特異解では，Hを (3.41) のように分解したH_2の時間微分が 0 という条件より最適特異入力が決定される．このとき，Hamiltinian=一定の条件より，制御対象の状態はある超曲面 $H_1 = const$ に拘束されたかのようになる．すなわち，特異解とは状態空間中である超曲面に拘束される解，つまりスライディングモード制御になっていることが分かる．これは (3.38) のような評価関数をとったときの最適解はスライディングモード制御となることを意味する．例えば，(3.42) のような 1 入力を持つ 2 次の制御対象に対して，評価関数を (3.43) のように定めたときの最適特異解は (3.44) と表わされる．ただし，gは 0 にはならないとする．

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ f(x_1, x_2) + g(x_1, x_2)u \end{bmatrix} \tag{3.42}$$

$$L = \int_0^{T_c} \left(ax_1^2 + x_2^2\right) dt \quad , \quad a > 0 \tag{3.43}$$

$$x_2^2 = ax_1^2 + const. \tag{3.44}$$

ここで示されたように，制御対象 (3.36) に対して，ニューラルネットワークを (3.39) のような評価関数に基づいて学習させることにより得られる制御系は，スライディングモード制御となる．スライディングモード制御を学習するニューラルネットワークとして，中間層が 1 層の階層構造型であり，活性化関数としては入力層では恒等関数，中間層ではシグモイド関数，出力層では飽和関数を用いるものがよいと考えられる．これは以下のような理由による．中間層の活性化関数としてシグモイド関数を用いることにより，学習後のニューラルネットワークは連続関数となることから，不連続なスライディングモード制御系の連続関数による近似が得られる．実際に，スライディングモード制御では入力のチャタリングを防ぐために切換面に境界層を設け，境界層の中では入力が連続的に変化するようにすることが多いので，このような連続関数による近似は実用の際に非常に好ましいものである．また，境界層の形状はシグモイド関数に非常に似ているが，この点も中間層の活性化関数としてシグモイド関数を用いるニューラルネットワークの利点の一つといえる．また，出力層で用いる飽和関数は制御入力の飽和を取り扱うためのものである．出力層に飽和関数を用いて学習を行なうことにより，学習において制御入力の飽和特性が考慮されることとなる．

(3.43) で表されている最適特異解となる曲線群のなかで，原点を通るものは原点を通る直線のみである．高次の制御対象であっても，(3.43) のような状態のみの二次形式で表される評価関数を用いる場合には，原点を通る最適特異解は状態空間中の平面である．このために，スライディングモード制御に関する研究において，線形な滑り面が一般的に用いられることは自然であると言える．しかし，二次形式以外など一般の評

価関数を考えることにより，滑り面も非線形 (曲面) に設計することができることも分かる．これはある種の非線形の滑り面の理論的な根拠 [38] となると考えられる．

3.6.3 スライディングモード制御系の学習アルゴリズム

本節では，ニューラルネットワークの学習によるスライディングモード制御系の構築の際に用いる学習アルゴリズムを考察する．第2章で示したように，勾配法に基づく学習アルゴリズムでは任意の Bang-Bang 解において結合度による評価関数の勾配が 0 となる．このため任意の Bang-Bang 解は学習の局所解となる．すなわち，勾配法に基づく学習アルゴリズムによる学習を用いる場合，学習の途中でニューラルネットワークが最適ではない任意の Bang-Bang 制御となると，そこで学習は終了となってしまう．しかし，既に述べたように，スライディングモード制御はある切換面で急激に入力を切換るものであり，Bang-Bang 解の一種といえる．このため，最適ではない Bang-Bang 解を学習の途中で経由することが多いと考えられる．よって，任意の Bang-Bang 解で学習が停止する勾配法に基づく学習アルゴリズムは，多くの局所解の存在のためにその性能を十分に発揮できない．

これに対して，Powell の共役方向法に基づく学習アルゴリズムでは評価関数の勾配を必要としないために，任意の Bang-Bang 解が学習の局所解となるという欠点が存在しない．このために，学習の途中でニューラルネットワークが最適でない任意の Bang-Bang 制御系となった場合でも，学習は停止することはなく，必要であれば学習はさらに進む．また，学習に必要な繰り返し数も少なく，短時間で解を見つけることもできるため，スライディングモード制御系の学習には Powell の共役方向法を用いる学習アルゴリズムが適していると考えられる．

ニューラルネットワークを用いたスライディングモード制御系のブロック図は図 3.1に示したものと同じである．ニューラルネットワークの学習では，評価関数は (3.39) のように離散時間のものを用いる．また，システム (3.36) も離散化して学習を行う．

3.6.4 ロバスト性に関する検討

滑り面に状態があるとき，すなわちスライディングモード状態における外乱の除去能力の高さのために，スライディングモード制御は高度なロバスト性を持つと言われている [35–37]．しかし，モデル化誤差などが存在するときにスライディングモード状態に移行できるかどうかという保証はされていない．このために，そのロバスト性の高さが発揮できるかどうかは明らかでない．また，入力の切換を行なうために制御系が非線形となるので，ロバスト性の定量的な評価としてよく用いられる H_∞ゲインを求めることができなかった．このために，スライディングモード制御系のロバスト性を定量化する方法がなかった．

3.6.3節で示したスライディングモード制御系の学習法は，基本的に2章で示したニューラルネットワークの学習による最適フィードバック制御系の学習と同一である．このために，これまでの研究 [35–37] により示されたスライディングモード制御系の高度なロバスト性から，学習後のニューラルネットワークのロバスト性は高いと期待されるだけであり，学習を行なう際にはロバスト性の定量的な評価はない．このような欠点は，3.3節および3.4節に示したロバスト性を定量的に保証する学習により解消される．また，ロバスト性を保証しない学習であっても，学習が完了したニューラルネットワークのロバスト性を評価することにより安全に制御を行うことができる．また，3.3節において示したロバスト性の定量的な解析方法は非線形制御系に対しても用いることができるため，スライディングモード状態に移行する前のロバスト性をも考慮することができ，従来のスライディングモード制御系の一般的な定量的なロバスト性評価方法としても用いることができる．

3.6.5 数値例

スライディングモード制御の例を示すには2次である制御対象の例を挙げるのがもっとも分かりやすい．このため制御対象のモデルを次のような2次の非線形システムとする．

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ (1-x_1^2)x_2 - x_1 + u + v \end{bmatrix} \tag{3.45}$$

$$|u| \leq 1.0 \tag{3.46}$$

ニューラルネットワークの学習を行なう際には，(3.45) のモデルをサンプリング時間幅 $h = 0.01$ で離散化を行った．

スライディングモード制御の学習

スライディングモード制御をニューラルネットワークに学習させるために評価関数を次のように定める．

$$J = \sum_{t=0}^{T} ax_1^2(t) + x_2^2(t) \tag{3.47}$$

(3.47) における a は正の定数パラメータである．ここでは中間層のユニット数が10であるネットワークを用いて学習を行なった例を示す．学習の際に，初期状態は $[0.5, 0]^T$ であるとした．$a = 0.5$ として学習したニューラルネットワークによる制御の結果を図3.17に示す．図3.17をみると，状態は理論解と同じ滑り面に拘束されながら原点へと滑って行くことが分かる．次に $a = 3$ と変更して学習を行なったニューラルネットワークによる制御の結果を図3.18に示す．このとき，状態は理論解と同じ滑り面に拘束されながら原点へと滑るのは図3.17と同じであるが，制御入力は滑り面で切り換わらなかった点が大きく異なっている．学

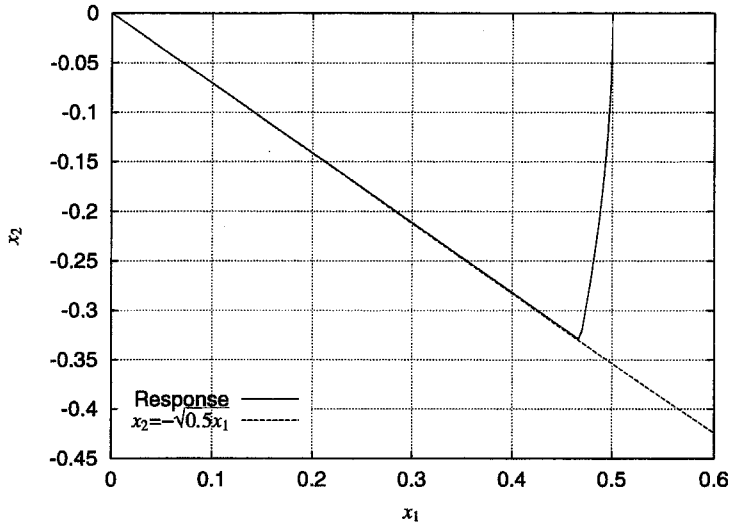


図 3.17: Transition of the state($a = 0.5$)

習された入力切換面を示すために，学習が完了したニューラルネットワークの出力が±1 の中間となる制御対象の状態 (x_1, x_2) を調べ，プロットしたものを図 3.19に示す．この図によると，入力切換面だけでなくその境界層がはっきり示されており，境界層を持った入力切換面をニューラルネットワークは学習していることが分かるが，入力切換面は滑り面と一致せず，滑り面を一部として含んだ非線形なものとなっていることが分かる．これは制御対象のモデルが非線形であるために生じる現象でなく，線形のモデルに対して設計した場合でも生じる現象である．この現象が起こる原因は入力の飽和特性である．スライディングモード制御では状態が滑り面に達したとき，制御入力を急激に切り換えることで状態を滑り面に拘束しようとするが，この際に過大な入力を要求することが多い．このために，制御入力に厳しい制約がある場合，滑り面に拘束することが困難となる．よって，入力に飽和特性があると入力切換面は一般に非線形となり，滑り面と入力切換面は完全に一致せず，滑り面は入力切換面の部分空間となる．従来のスライディングモード制御理論では，このような入力の切り換えは全く考えられていない．また，最適制御理論を用いてもこのような制御系の設計は困難であったが，本手法によると容易にスライディングモード制御系を設計できる．

ロバスト性の測定

学習に用いる評価関数 (3.47) に現れる a を $0 < a \leq 30$ と変化させて学習を行ない，それぞれのニューラルネットワークのロバスト性を 3.3節の方法を用いて測定した．このとき，ロバスト性の評価のための学

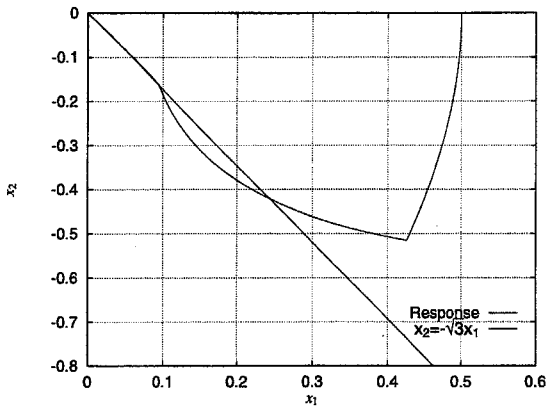


図 3.18: Transition of the state($a = 3$)

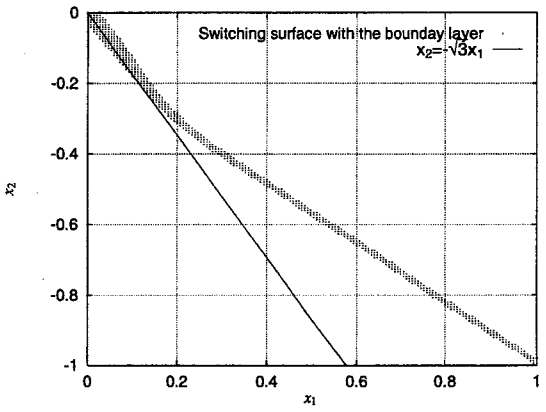


図 3.19: Switching surface and boundary layer obtained by the trained neural network($a = 3$)

習に用いた評価関数は (3.48) とした.

$$J = \sum_{t=0}^{T} \left[a x_1^2(t) + x_2^2(t) - \gamma^2 v^2(t) \right] \tag{3.48}$$

図 3.20に a とそのときのロバスト性をγを示す. 図 3.20によると a が小さいほど高いロバスト性を持っていることが分かる. これは評価関数 (3.47) は a を大きくすると早く原点に収束させることを目的としたものとなり, 速い応答性持つことを要求するためにロバスト性が犠牲になるからである. よって, a の増加に伴い, γは単調に増加すると考えられる. しかし, 図 3.20によると, a が 10 程度となった辺りでγが振動しており, a を増加させてもロバスト性は低下せずに向上してしまう場合があった. これは明らかにロバスト性と制御性能の関係から矛盾した関係となっている. この原因を探るために, 中間層のユニット数を変化させてスライディングモード制御系の学習を行ったニューラルネットワークのロバスト性を測定した. $a = 6$ および $a = 8$ のとき, 中間層のユニット数 Nを 1 から 15 まで変更したときの学習後の評価関数とγ(ロバスト性) を図 3.21, 3.22, 3.21, 3.24に示す. 図 3.21, および図 3.23を見ると, ある数以上の中間ユニットがあれば学習後の評価関数はほとんど変らないことが分かる. しかし, 図 3.22, および図 3.24によると, ロバスト性の大きさは中間層ユニット数 Nにより激しく変動していることが分かる. また, これらの図より, その変動に規則性を見いだすことができなかった. この結果は [41] で示したように, 第 2章のように不確かさを考慮せずに, ニューラルネットワークの学習により最適フィードバック制御系の構築を行ったときには, そのロバスト性は保証できず, 期待よりロバスト性が低い制御系が得られる場合があることを示しており, 学習においてロバスト性を考慮することが必要であることを意味する.

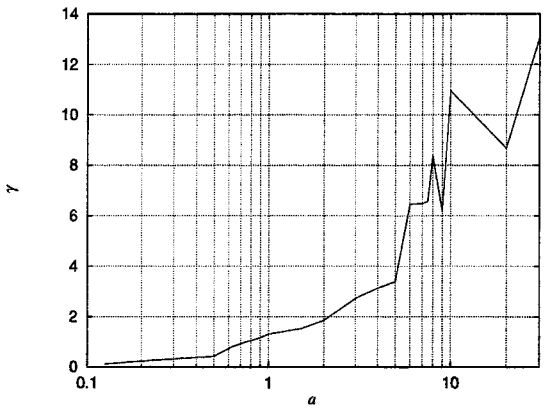


図 3.20: Relationship between a and the robustness of trained neural networks

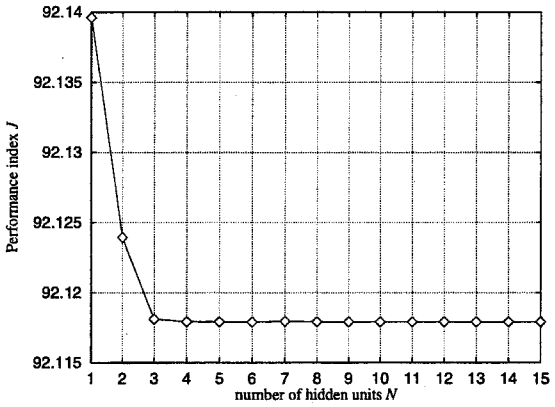


図 3.21: Relationship between the number of hidden units and performance of trained neural networks ($a = 6$)

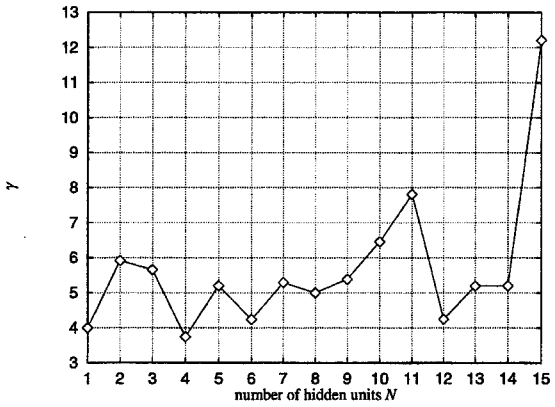


図 3.22: Relationship between the number of hidden units and robustness of trained neural networks ($a = 6$)

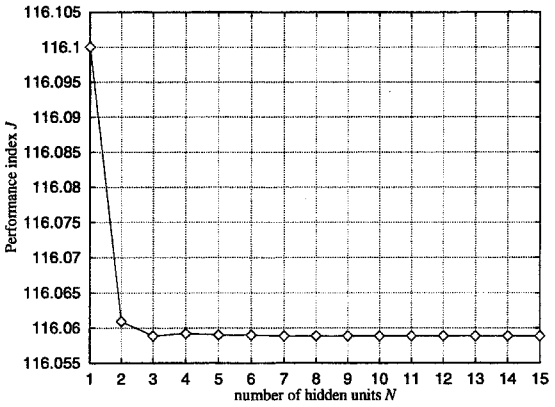


図 3.23: Relationship between the number of hidden units and performance($a = 8$)

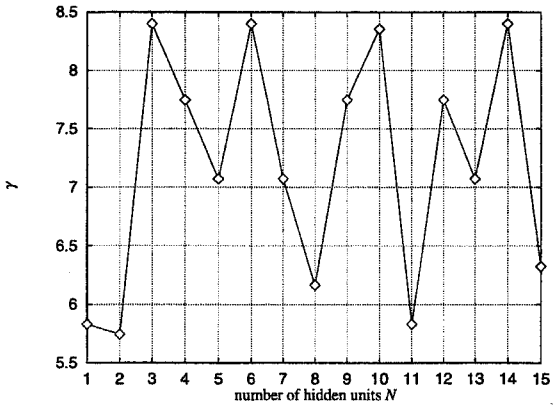


図 3.24: Relationship between the number of hidden units and robustness($a = 6$)

ロバスト制御系の学習

図3.20で見られたγの振動は，複数の初期条件を用いて学習を行うことにより改善されるが，やはりγの振動が確認された．また，初期条件を複数にして学習を行っても，制御系のロバスト性を保証した制御系にはならないので不十分である．そこでロバスト性の定量的な保証のために，3.3.2節で示した学習を行った．学習の際に用いた初期条件は$[0.5, 0]^T$と$[-0.5, 0]^T$を用いた．このときのaとγ(ロバスト性) の関係を図3.25に示す．このとき，図3.20と異なり，γの振動は見られなくなった．また，aの増加に対して単調に増加することでロバスト性が単調に減少していることが分かる．．さらに，各aにおいてはロバスト性が向上していることが分かった．実際に，ロバスト性が大きく向上したaでは，最適特異解によりスライディングモード制御系を学習したニューラルネットワークのロバスト性が低くなっていたと考えられる．また，aが小さいときは概してロバスト性の向上は顕著ではなかった．これは小さいaを用いて最適フィードバック制御系の学習を行ったニューラルネットワークはそもそも十分にロバスト性が高く，ロバスト性に優れたスライディングモード制御系を学習できていることを示している．しかし，aを小さくすると，制御系の応答は遅くなり，その結果として制御性能が低くなるため，制御性能が保守的な制御系となっている．

3.3.2節で示した学習によりニューラルネットワークのロバスト性が向上したことを確認するために，$a = 4$のときにスライディングモード制御系を学習したもの [**A**] とロバスト性を考慮したもの ($\gamma = 8.5$ として学習)[**B**] を比較する．その際，[**A**] にとって最悪の不確かさ ($\gamma = 13.2$ のとき) がともに存在するとしてシミュレーションを行った結果を，図3.26に示す．明らかにロバスト性を考慮した学習によって，ロバスト性の向上が達成されていることが分かる．

非線形滑り面の設計

ここでは，二次形式以外の一般の評価関数を設定し，非線形滑り面の設計を行う．ここでは簡単な例として，(3.49) および，(3.50) のように評価関数を設定し学習を行う．

$$J = \sum_{t=0}^{T} x_1^2(t) + x_2^4(t) \tag{3.49}$$

$$J = \sum_{t=0}^{T} x_1^4(t) + x_2^2(t) \tag{3.50}$$

それぞれの場合の，最適特異解は (3.51)，(3.52) となる．

$$x_1^2 = 3x_2^4 + const. \tag{3.51}$$

$$x_1^4 = x_2^2 + const. \tag{3.52}$$

評価関数を (3.49) および, (3.50) として学習を行なったニューラルネットワークによる制御結果を図 3.27, 3.28 に示す. これらの図より, ニューラルネットワークによると非線形滑り面を持つスライディングモード制御系の設計も容易であることが分かる.

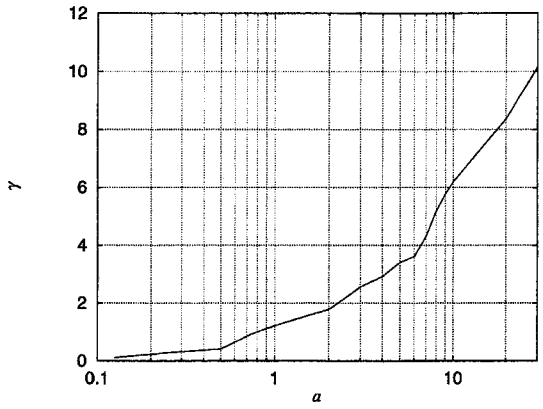


図 3.25: Robustness of neural networks trained by the proposed method

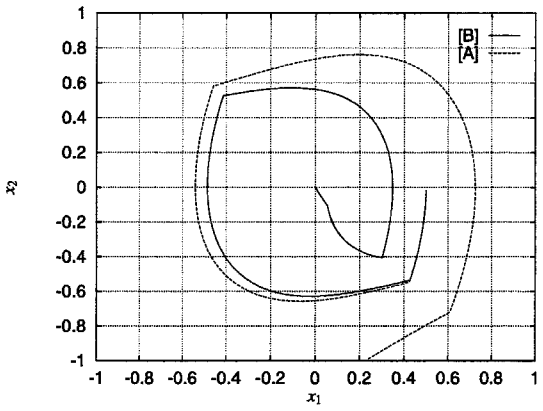


図 3.26: Responses of the worst system including uncertainties

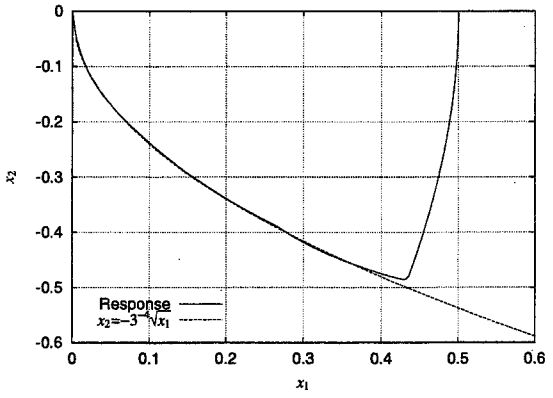


図 3.27: An example of designing a nonlinear sliding surface

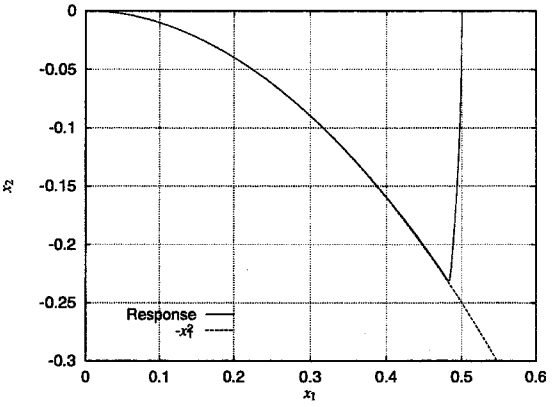


図 3.28: An example of designing a nonlinear sliding surface

3.7 複数の不確かさに対するロバスト制御系の学習

3.7.1 複数の不確かさに対するロバスト性

これまでに述べたように，制御対象にはモデル化誤差や外乱などに代表される不確かさが存在するために，不確かさに対してロバストな制御系を構築することが好ましい．制御対象になんらかの不確かさが存在しても，システムを安定に保つことができることをロバスト安定であるといい，制御系の重要な特性である．制御系のロバスト安定の指標としてよく用いられるものに，ロバスト安定性とロバスト制御性能がある．ロバスト安定性は公称対象における外乱などの不確かさの制御性能や安定性に対する影響度として評価され，制御系のロバスト性を評価する重要な特性である．ロバスト制御性能とは変動が生じた対象における外乱などの不確かさに対する制御性能への影響で表される．明らかなように，ロバスト制御性能の方がロバスト安定に比べると厳しい指標である．しかし，制御系に求められるロバスト性とはそもそもはロバスト制御性能のことであり，ロバスト安定性を保証するだけで十分であるとはいえないことが多い．なぜなら，制御対象が公称モデルであることは期待することはできないので．公称モデルに対する外乱の影響度を軽減することはあまり意味がないことが多いのに対して，擾乱が加わったモデルに対する外乱の影響度を考慮するロバスト制御性能のは実際に即しているからでる．しかし，H_∞制御における混合感度問題などでは，ロバスト性をロバスト安定性を用いて評価しており，ロバスト制御性能を取り扱っていない [30]．ロバスト制御性能を評価するには複数の不確かさを考慮する必要があるのだが，[30] の方法では単一の非構造的な不確かさしか取り扱うことができないからである．また，[30] の方法では複数の不確さが存在する場合には単一の不確かさに変換する必要があるので，変換後の不確かさの範囲が大きくなり，このためにロバスト性に対する評価が保守的となることが多かった．そこで，構造的な不確かさや複数の不確かさを取り扱うためにμ解析/設計などが考え出されたが，線形な対象しか取り扱うことができなかった．また，H_∞制御を非線形な領域まで拡張する試みもなされているが [42]，やはり単一な非構造的な変動しか考慮することができず，ロバスト制御性能を保証することはできなかった．

3.3節や3.4節において非構造的な変動，あるいは構造的な変動に対してロバストな制御系をニューラルネットワークの学習により構築する方法を提案した．これらの方法によると，ニューラルネットワークを制御系に応用する従来の研究において不可能であったネットワークのロバスト性の定量的な保証が可能となるだけでなく，任意の制御系のロバスト性を解析し，定量的に評価することも可能となった．しかし，3.3節や3.4節では，不確かさを単一の構造的あるいは非構造的な不確かさと限定したので，複数の不確かさは取り扱うことはできなかった．このため，これらの方法ではロバスト性の指標としてはロバスト安定性を用いており，ロバスト制御性能を定量的に保証することはできないので，制御系に対して本来要求される仕様

を満たす制御系を学習することが可能であるとはいえなかった．また，不確かさが完全に構造化されるほどの情報が存在することは実際には少なく，不確かさの一部のみが構造化されることの方が多い．よって，制御対象には一般的に非構造的な不確かさと構造的な不確かさの両方が混在するが，3.3節や3.4節の方法ではこのような系を取り扱うことはできなかった．そこで，本節では複数の不確かさが同時に存在する制御対象を考慮し，そのロバスト性，特にロバスト制御性能が定量的に保証された制御系をニューラルネットワークの学習により構築する方法を提案する．

3.7.2 対象とする問題

本節で対象とする制御対象は，不確かさが存在する離散時間システムであり，(3.53) で表されるとする

$$\boldsymbol{x}(t+1) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t), \boldsymbol{\alpha}(t)) \tag{3.53}$$

ここで，$\boldsymbol{x}$は対象の状態変数であり，全て測定可能とする．$\boldsymbol{u}$は制御入力を表し，(3.54) のように制約されているとする．

$$\boldsymbol{u} \in \boldsymbol{U} \tag{3.54}$$

$\boldsymbol{\alpha}$は互いに独立な不確かなパラメータを並べたベクトルであり，構造的な不確かさを表す．定量的な評価のため，$\boldsymbol{\alpha}$の存在する範囲を既知な閉集合Δ内であると仮定する．

$$\boldsymbol{\alpha}(t) \in \Delta \quad (^{\forall}t \geq 0) \tag{3.55}$$

また，$\boldsymbol{v}$は非構造的な不確かさを表している．

本節においても，不確かさを含む制御対象に対して，原点を安定に保つ状態フィードバック制御系をニューラルネットワークの学習により設計することを目的とする．設計を行う制御系のブロック線図を図3.1に示す．ここでは階層構造型ニューラルネットワークを用い，静的な状態フィードバック制御系を構築する．このため，制御対象の原点は，不確かさが存在しても平衡点であると仮定する．

3.7.3 複数の不確かさに対する制御系の学習法

複数の不確かさが同時に存在する組み合わせには，

[A] 構造的な不確かさのみからなる組合せ

[B] 非構造的な不確かさのみからなる組合せ

[C] 構造的と非構造的な不確かさが複合する組合せ

の3種類が存在する．以下に，それぞれの場合に対する学習法を述べる．

[A] に対しては，3.4節で示した方法をそのまま用いることができる．その際，不確かさに関する情報が十分に利用されるので，保守性も問題とならない．

[B] に対しては，3.3節で示した方法が適用できる．しかし，この場合は保守性が問題になる．不確かさが構造的な情報を持たないため，複合されることにより実際に存在し得ないような不確かさまでが考慮される可能性が大きくなる．また，3.3節で示した方法では評価関数 (3.7) にあらわれるように，ロバスト性を一つのゲインγで評価している点も問題である．対象に含まれる不確かさを$\boldsymbol{v}_i(i=1,2,\cdots)$とすると，

$$J=\sum_{t=0}^{T}||\boldsymbol{z}(t)||^2-\sum_{i}^{T}\gamma_i^2||\boldsymbol{v}_i(t)||^2 \tag{3.56}$$

のように評価関数を変更することにより，効率よくロバスト性を配分することができるため，保守性を低減することができる．このことは，ロバスト性の解析においても同様である．

[C] に対しては，既に提案されている方法をそれぞれ単独に用いることでは対応することができない．そこで [C] に対してロバスト制御系を構築する方法を構築する必要がある．

3.3節や 3.4節で示した方法は，構造的な不確かさ，あるいは非構造的な不確かさのいずれかしか考慮することができない．(3.53) のようにそれぞれの不確かさが同時に存在する対象に対して，3.3節の方法をそのまま応用すると，構造的な不確かさの情報を十分に利用することができず，保守的な結果になる．また，3.4節の方法ではロバスト性を保証することすらできない．よって，これらの方法はそのまま利用することはできないが，これらは「制御系にとって最悪な不確かさが存在すると考えて，その時の制御性能を考慮して設計する」という共通の指針に基づいて導出された学習法である．再び，この考えに基づくことにより，3.3，3.4節の方法より，複合した不確かさ存在する対象に対して，ロバストな制御系を学習することができる新しい方法を導くことができる．

新しい学習方法では，3.3節と同様に，フィードバック制御器としてのネットワークと，不確かさを表すネットワークが必要である．複数の非構造的な不確かさが存在する場合は，保守性の低減化のためにすでに示したようにそれぞれを別々に取り扱うことが好ましいが，ここでは議論の簡単化のために，非構造的な不確かさを一つにまとめて説明をする．以下では，構造的な不確かさが時変，あるいは時不変な場合それぞれについて説明する．

時変な構造的不確かさを含む場合

構造的な不確かさに時変な要素が含まれる場合は，その時間的な変動も制御系を学習させる際に考慮しなければならない．このために，学習には図 3.29のようなブロック図を用いる．ここで，不確かさを模擬するニューラルネットワークは$\boldsymbol{v}(t)$と$\boldsymbol{\alpha}(t)$を出力する．非構造的な不確かさを定量的に評価するために，3.3節と同様に$\boldsymbol{v}$から$\boldsymbol{z}$への L_2ゲインに相当するγを含んだ評価関数を用いる．さらに，構造的な不確かさのた

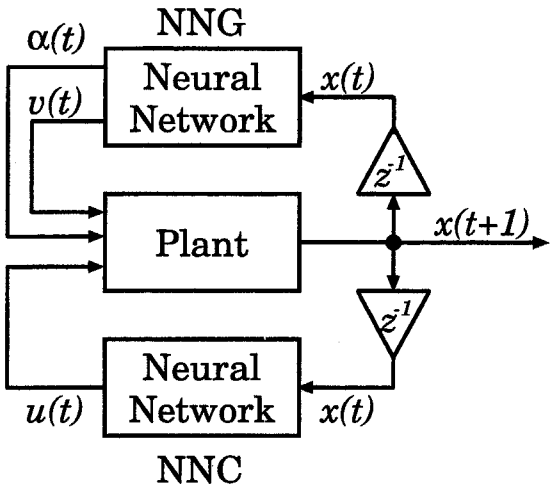


図 3.29: Block diagram for training robust controllers against multiple uncertainties

めにロバスト制御系の学習は次の最適化問題に帰着することができる.

$$\min_{\boldsymbol{u}(t)} \max_{\boldsymbol{\alpha}(t)\in\Delta} \max_{\boldsymbol{v}(t)} \sum_{t=0}^{T} ||\boldsymbol{z}(t)||^2 - \gamma^2||\boldsymbol{v}(t)||^2 \tag{3.57}$$

学習後のネットワークのロバスト性は不確かなパラメータ$\boldsymbol{\alpha}(t)$の存在範囲やγによって定量的に保証される. また, 本手法によるとロバスト安定性だけでなく, ロバスト制御性能を定量的に評価することが可能になる.

ニューラルネットワークの学習はロバスト制御系設計問題だけでなく, ロバスト性解析にも本手法を応用することができる. ロバスト性の解析を行なう場合には, 制御器を固定であり, 不確かさを表したネットワークのみ学習を行う. このとき, ニューラルネットワークの学習は次の最適化問題に帰着される.

$$\max_{\boldsymbol{\alpha}(t)\in\Delta} \max_{\boldsymbol{v}(t)} \sum_{t=0}^{T} ||\boldsymbol{z}(t)||^2 - \gamma^2||\boldsymbol{v}(t)||^2 \tag{3.58}$$

時不変な構造的不確かさを含まない場合

構造的な不確かさに時変な要素を含まないときは, 不確定パラメータの動的な変化を考慮しなくてもよいので, 学習には図3.2と同じシステムを用いることができる. 時変な構造的不確かさを考慮したときと同様に考えると, ロバスト制御系設計問題は最適化問題 (3.59) に帰着され, ロバスト性解析問題は最適化問題 (3.60) に帰着される.

$$\min_{\boldsymbol{u}(t)} \max_{\boldsymbol{\alpha}\in\Delta} \max_{\boldsymbol{v}(t)} \sum_{t=0}^{T} ||\boldsymbol{z}(t)||^2 - \gamma^2||\boldsymbol{v}(t)||^2 \tag{3.59}$$

$$\max_{\boldsymbol{\alpha}\in\Delta} \max_{\boldsymbol{v}(t)} \sum_{t=0}^{T} ||\boldsymbol{z}(t)||^2 - \gamma^2||\boldsymbol{v}(t)||^2 \tag{3.60}$$

時不変である構造的な不確かさは，時変なものより存在範囲が小さいことは明らかであるが，このとき制御系に要求されるロバスト性は小さくなる．前節のように，不確かさαを時変であるとすると，その最悪な時間変動は一般的にBang-Bang解である．このために，構造的な不確かさの時間変化がシステムの応答に比べて十分にゆっくりとしたものである場合に対して，前節の手法をとると保守的な結果になりやすい．このように時間的な変化がゆっくりとしたものであるときには，時不変であると仮定し，学習させるほうが制御結果が好ましいものになる．

3.7.4 数値例

提案手法の有効性を示すために，簡単な数値シミュレーション例を示す．

複数の非構造的な不確かさが存在する場合

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} x_1(t) + h(x_2(t) + v_1(t)) \\ x_2(t) + h\{\left(1.0 - x_1^2(t)\right) x_2(t) - x_1(t) + u(t) + v_2(t)\} \end{bmatrix} \tag{3.61}$$

$$|u(t)| \leq 1.0 \tag{3.62}$$

$$\boldsymbol{x}_0 \in \Omega\{\boldsymbol{x} : x_1^2 + x_2^2 \leq 0.5^2\} \tag{3.63}$$

ここで$h = 0.01$であり，$v_1(t)$と$v_2(t)$は非構造的な不確かさである．これらは3.4節のように考えて単一の非構造的な不確かさとして取り扱うこともできるが，ここでは複数の非構造的な不確かさとして取り扱う．また，初期条件は(3.63)で与えられ，ある固定点ではないとした．ニューラルネットワークの学習を用いて制御系を設計する研究では，その制御対象の初期条件は固定点であることが多く，実際の初期条件との違いはネットワークの汎化性に依存させ，あまり考慮していないことが多いが，それは3.5節で述べたように不十分である．実際に，初期条件はある領域の含まれることは分かっているが，一定と定まっていないことの方が多い．3.5節で示したように，初期条件の曖昧さは時不変なパラメータの不確かさとして扱うことができる．本節で示す例では，初期条件の曖昧さはニューラルネットワークの学習に含めて考慮することとする．

ここでは，提案手法によりロバスト性解析を行なった例を示す．制御器としては，初期条件を(3.63)内のある一点に定めて，次の評価関数を最小にするような最適フィードバック制御系をニューラルネットワークに学習させたものを用いた[32]．

$$J = \sum_{t=0}^{1000} x_1^2(t) + x_2^2(t) + u^2(t) \tag{3.64}$$

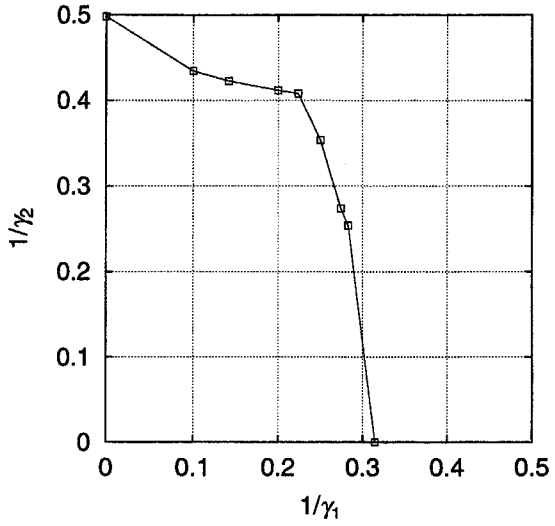


図 3.30: Result of the analysis of the robustness

このニューラルネットワークのロバスト性は定量的に保証されていないため，提案手法により，この制御系のロバスト性の解析を行なった．このとき，学習は次の最適化問題となる．

$$\max_{\boldsymbol{x}(0)\in\Omega}\max_{v_1,v_2}\left[J-\sum_{t=0}^{1000}\gamma_1^2v_1^2(t)+\gamma_2^2v_2^2(t)\right] \tag{3.65}$$

ニューラルネットワークの学習より，安定であるγ_1，γ_2の範囲を求めた結果を図 3.30に示す．図 3.30では$1/\gamma_1$および$1/\gamma_2$を用いたが，これらの値は大きいほどロバスト性が大きいことを意味することに注意する．非構造的な不確かさを一つにまとめることによって，ロバスト性を評価すると$1/\gamma=0.27$であった．この結果と図 3.30を比べると，提案手法によりロバスト性の大きさをさらに細かく解析することが可能であり，解析の保守性が低減されることが分かる．

非構造的，および構造的な不確かさ共に存在する場合

次に，構造的な不確かさと非構造的な不確かさをともに含む例として (3.66) で表される制御対象を考える．

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} x_1(t)+h\cdot x_2(t) \\ x_2(t)+h\{c\cdot\left(1.0-x_1^2(t)\right)x_2(t)-k\cdot x_1(t)+u(t)+v(t)\} \end{bmatrix} \tag{3.66}$$

ここで$h=0.01$であり，cおよびkは不確かなパラメータで，構造的な不確かさをあらわす．また，$v(t)$は非構造的な不確かさである．入力の拘束条件と初期条件は (3.62)，(3.63) と同じとした．評価出力$\boldsymbol{z}(t)$は次のように定義する．

$$\boldsymbol{z}(t)=[x_1(t),x_2(t),u(t)]^T \tag{3.67}$$

ここで，不確かなパラメータ c, kは時不変であるとし，それぞれ公称値は 1.0 であるが±10%の偏差を持つと仮定する．比較のために，3.3節で説明した構造的不確かさのみを考慮する方法で学習を行なったニューラルネットワーク ([B]) と，提案手法により学習を行なったニューラルネットワーク ([A]) それぞれについて調べた結果を示す．ここで，[A] の学習を行なうときに用いたγは$\sqrt{5.00}$とした．構造的な不確かさに対して [B] のロバスト性は保証されているが，構造的な不確かさにさらに非構造的な不確かさが加わって存在する場合には，そのロバスト性は全く予測することはできない．そこで，本稿で提案した手法により，[B] のロバスト性を解析すると，$\gamma = \sqrt{5.50}$であることがわかった．γの値を比較すると，提案手法で学習させたニューラルネットワークの方が高いロバスト性をもつことがわかる．

それぞれのニューラルネットワークによる，公称制御対象の応答例を図 3.31に示す．提案手法で学習を行なったネットワークの方が少し応答が遅くなっているが，これは非構造的な不確かさの存在を考慮したためである．それぞれの制御系にとって最悪の構造的な不確かさのみが生じたとしたときの応答を図 3.32に示す．最悪な構造的不確かさが存在にもかかわらず，それぞれの応答は公称制御対象のものと比べて大きく変化せず，良好な結果を得ている．しかし，最悪な非構造的な不確かさまで存在するときの結果は大きく異なる．図 3.33に最悪な不確かさが存在するときのそれぞれの応答 ($\gamma = \sqrt{5.49}$のときの最悪な不確かさが存在) を示す．$\gamma < \sqrt{5.50}$としたときの最悪な不確かさが存在する場合，[B] はすでに安定ではないが，[A] は安定であるはずである．図 3.33より明らかなように，[A] は最悪な不確かさの存在にも関わらず系を安定に保つことができているのに対し，[B] ではすでに安定を保つことができていない．また，提案手法による学習をさらに進めることにより，$\gamma = \sqrt{4.70}$までロバスト性を向上させることができた．

表 3.1に，従来法を用いて学習を行なったネットワークのロバスト性の解析結果，および提案手法によりロバスト性をどこまで向上できるかを示した．構造化変動のみを考慮する方法では，構造化変動が時変であるとした方がvに対するロバスト性が低くなっている．これは時間的変動を考慮することにより，vへのロバスト性が減少するからである．一般に，構造的な不確かさが時変であると仮定してロバスト性解析を行なうより，時不変であると仮定して解析を行なったものの方がロバスト性が高く評価される．しかし，提案手法を用いて得られるロバスト性の最大値は同じであった．これは，最悪なパラメータ変動が時間的に一定となったためであり，矛盾する結果ではない．表 3.1は提案手法により，ロバスト性を保証するだけでなく，従来法より向上させることができることを示している．

Table 3.1: Comparison of the robustness

	構造化変動が時変	構造化変動が時不変
構造的な不確かさのみ考慮	$\sqrt{5.89}$	$\sqrt{5.49}$
提案手法	$\sqrt{4.70}$	$\sqrt{4.70}$

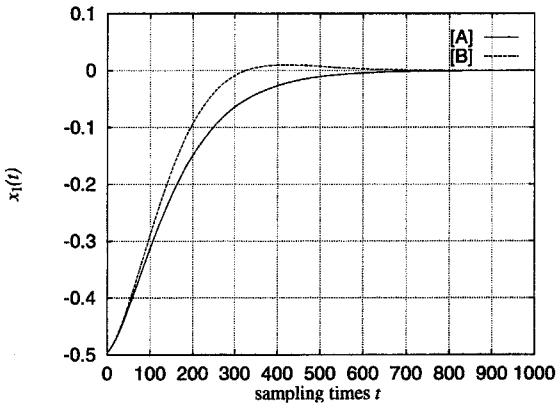


図 3.31: Nominal response of each neural network

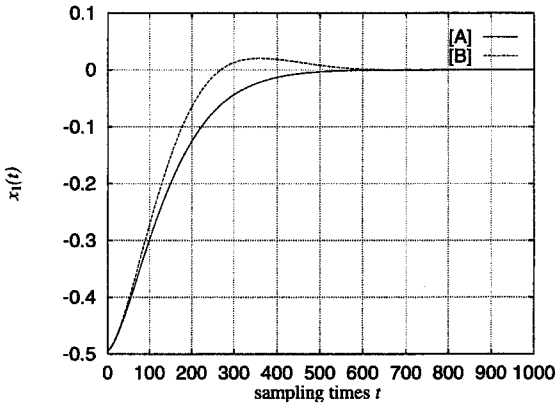


図 3.32: Responses of the system where only the worst stucuctured uncertainties exist

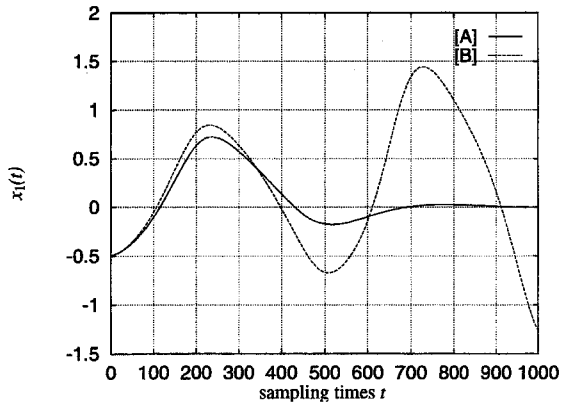


図 3.33: The worst response of each network

3.8 結言

本章では，ニューラルネットワークを用いてロバストな制御系を学習する基本的方法を提案し，数値計算によりそれらの有効性を確認した．学習後のネットワークのロバスト性の定量的な評価が可能であることが，提案方法と従来の研究と大きく異なる点である．また，任意の制御系のロバスト性を定量的に評価することが，ニューラルネットワークの学習により可能であることも示した．

3.3節では非構造的な不確かさを含む制御対象を取り扱った．この場合のロバスト制御系の学習は，図3.2のように二つのニューラルネットワークの競合となり，不確かさ$\boldsymbol{v}$から評価出力$\boldsymbol{z}$へのH_∞ゲインγを含んだ評価関数(3.7)を用いることにより，不確かさに対するロバスト性を定量的に保証された制御系を構築することができる．また，図3.2において制御系を学習するネットワークを固定，あるいは固定した任意の制御器として，最悪の不確かさを学習するネットワークのみを学習させることにより，ニューラルネットワークによる制御系をはじめとして任意の制御系のロバスト性を定量的に評価することができる．しかし，ロバスト制御系の学習およびロバスト性の評価を行う際には最小のγを求める効率的な方法が存在しないために，繰り返し計算を行なう必要がある．ロバスト制御系の学習，ロバスト性の解析のための学習，いずれの学習でも，γを大きな値から徐々に小さくしていくと，ある値を境に解の分岐のようが現象が確認され，評価関数の大きさがγの変化により突然変化し，制御系はそこで不安定になるのでγによる安定・不安定の判定は容易であるが，最小のγを正確に求めるには多くの繰り返し計算が必要である．しかし，最小のγを求めることは実際にはあまり多くないと思われる．あるγを用いて提案手法により学習をおこなったネットワークは，学習に用いたγが学習後のネットワークのロバスト性の最小値を示している．同様に，ロバスト性の解析を行なう場合でもあるγにより学習を行なった結果，その制御系が安定であれば，その制御系のロバスト性はすくなくとも$1/\gamma$より大きいことが分かる．このためにいずれの学習であっても，ロバスト性は安全側に余裕を持った評価がされているために十分と考えられるからである．また，小さなγを用いて学習を行なうこと，すなわちロバスト性を高めようとすることは制御系の保守性の問題を引き起すので実用的にあまり好ましい制御系にならないことが多い．以上のような理由から，γの最適化があまり頻繁に行われるとは考えられない．しかし，ロバスト性の評価の場合では，安全マージンを限りなく少なくする，すなわち可能な限り正確な最小のγを用いてロバスト性を評価したいときも多いと考えられるが，その場合はγの繰り返し計算に頼らざるを得ない．しかし，Powellの共役方向法を学習アルゴリズムとして用いることにより高速な学習を行なうことが可能である．実際に，本論文で用いた数値例では，わずか数回の繰り返し学習により，指定したγのロバスト性を持つかどうかが判定できた．この際の計算時間はPentium II-400MHzのPCで3分程度であり，最新のCPUを使えばさらに所要時間は短縮することが可能である．このように高速な学習が可能であるので，最悪の不確かさを探索に，ネットワークの初期点による局所解を防ぐ目的

で複数の初期ネットワークから学習を行なうとしても，十分に実用に耐えるものである．ロバスト制御系の学習では，やはり最適フィードバック制御系の学習に比べて，学習の計算に多くの時間を必要とするが，ここでもPowellの共役方向法を学習アルゴリズムとして採用することによりプラントのヤコビアンの計算を省くことが可能など高速化をが可能である．本研究で行った計算では十分な回数の学習を繰り返しているため，Pentium II-400MHzのPCを用いると6時間ほど時間を必要としたが，実際には数回の繰り返しで学習が完了することが多かったため，より短い時間で学習が完了することが可能であると考えられる．この学習方法は不確かさに対する先験的情報が無い場合に非常に有効であるが，学習で得られる制御系は保守的となることが多い．これは不確かさに対する情報がないために，存在し得ない不確かさまで考慮される可能性があるからである．このような難点を解消するために，不確かさに関する情報を利用し存在しえない不確かさを学習において考慮されることを防がねばならない．

3.4節において，3.3節において問題となった制御系の保守性を解消するために，構造的な不確かさを伴う制御対象に対するロバスト制御系の学習法を示した．不確かさの情報を有効に利用できるために，3.4節の学習法で得られる制御系のほうが制御性能は好ましいものとなり，学習により適切な大きさのロバスト性が確保される．構造的な不確かさはパラメータの不確かさに帰着されるが，本手法では，不確かなパラメータが時変であるときには，最悪なパラメータ変動はBang-Bang解のように急激な切換を伴うものとなり，不確かなパラメータが時不変であるときには，学習に用いる評価関数が微分不可能な関数となった．このために2.3節において示した勾配法に基づく学習アルゴリズムを適用することが難しい．しかし，2.4節において提案したPowellの共役方向法に基づく学習アルゴリズムでは，微分不可能である関数を学習の評価関数として用いることができるため，構造的な不確かさを伴う対象に対するロバスト制御系の学習が可能となる．Powellの共役方向法に基づく学習アルゴリズムは，構造的な不確かさを伴う対象に対するロバストな制御系の学習には不可欠である．

3.6節では，最適制御問題における特異解に着目することにより，代表的な非線形ロバスト制御系であるスライディングモード制御系をニューラルネットワークの学習を用いて設計する方法を示した．従来のスライディングモード制御系はスライディングモードにおける外乱除去性のためにロバスト性が高いと言われていたが，入力に厳しい制限がある場合にはスライディングモードに移行することができないことが多いことが知られおり，実用に際して大きな問題点となっていた．しかし，3.6節で提案するスライディングモード制御系の学習法により入力の制限を考慮することが可能となった．従来のスライディングモード制御系の設計方法では，まず最初に滑り面の決定を行ない，入力切換面は滑り面と全く同一なものとする．しかし，提案手法により学習された制御系では一般に入力切換面は滑り面と同一とはならず，制御入力の利用に厳しい制約があると，線形な滑り面を設計しても，入力切換面は状態空間中で非線形となっていること

が3.6節の結果より示された従来法では，このような入力切換面の設計は困難であったが，ニューラルネットワークの学習により簡単に設計が可能となる．また，中間ユニットの活性化関数がシグモイド関数であるニューラルネットワークを用いることにより，制御入力の切換は不連続なものとならず，境界層により連続化されたスライディングモード制御系が得られる．境界層による制御入力の連続化は入力のチャタリング防止に役立ち，高周波領域のモデル化されていないダイナミクスの励起を防ぐことができるので，従来のスライディングモード制御系でも行われている．しかし，従来の設計法では境界層の厚さなどを適切に定めることが難しかったが，ニューラルネットワークの学習にによると適切な厚さの境界層を獲得することができる．しかし，特異解を学習する方法は2章で示した最適フィードバック制御系の学習と同じであるため，学習後のネットワークのロバスト性は保証されないが数値例により学習後のネットワークは高いロバスト性を持つことが示された．しかし，学習が完了したニューラルネットワークが期待されるロバスト性を持っていない場合があることも同時に明らかとなった．そこで，中間層のユニット数を様々に変更し，それぞれ学習を行ったネットワークのロバスト性の確認を行ったところ，図3.22～3.21 のような結果が得られたが，ロバスト性を定量的に保証することができない学習方法では，学習後のロバスト性を予測することは困難であることが分かった．しかし，3.3節や3.4節の学習法を併用して，スライディングモード制御系の学習を行なうことが可能であるので，スライディングモードに到達する以前の制御も含めて，ロバスト性を定量的に保証されたスライディングモード制御系が構築可能となった．

3.3節や3.4節では，単一の非構造あるいは構造的な不確かさが存在する制御対象のみを考慮していた．しかし，実際には複数の不確かさが同時に存在することや，不確かさの一部のみに関する知識があるのみであるために部分的に構造化されている場合が多い．3.3節や3.4節で示した方法により学習を行ったネットワークは，複数の不確かさが存在する場合のロバスト性は全く保証されず，その予測をすることもできない．このように不確かさを適切な形で顧慮した制御系設計や，ロバスト性の定量的な解析を行うには複数の不確かさを取り扱う方法が必要であった．また，3.3節や3.4節ではロバスト性の定量的な評価にはロバスト安定性を用いていたが，これは十分と言えない場合も多く，より実用的なロバスト制御性能を考慮する方法が必要であった．このため，3.7節では，複数の不確かさが同時に存在する系に対し，ニューラルネットワークの学習によりロバスト制御系を構築する方法を示した．3.3節や3.4節で示した学習方法は，最悪な不確かさが存在するとして最悪な制御性能を改善するという共通の考えにより導かれたものであったために，容易に統合することが可能であり，適切な不確かさを考慮する方法へと発展させることができた．提案した手法では複数の不確かさを取り扱うことができるため，ロバスト性をロバスト安定性だけでなく，ロバスト制御性能を用いて定量的に保証することが可能である．また，ロバスト性の評価にも応用することができ，任意の制御系のロバスト制御性能を定量的に行なうことも可能となる．

ニューラルネットワークの学習は局所解に陥ってしまうことがあるが，本章で示した学習では制御系のネットワークの局所解よりも不確かさを表すネットワークの局所解の方が重大な問題となる．これはロバスト性を誤って高く評価してしまう可能性を持つからである．このために，本研究で行った数値例では不確かさを模擬するネットワークを複数の異なる初期状態から学習させることにより防いでいるが，効率などの面から言っても十分とはいえないと考えられる．行なった数値計算の結果などから，不確かさを表すネットワークでは集中的な探索による学習法よりも探索空間を多様な検索を行う学習法が必要であると思われるが，この件に関しては今後の課題といえる．多様な探索では遺伝的アルゴリズムの利用やタブ探索 [43,44] の研究があるが，結合度を量子化する必要があるために問題が大規模化することが多いのでニューラルネットワークの学習にそのまま応用することはできない．また，本章で示した方法はロバストな状態推定器や動的な制御器を学習する方法に拡張することが可能であると考えられる．

第 4 章

無人機の自律飛行制御系構築への応用

4.1 緒言

近年，無人機 (Unmanned Aerial Vehicle:UAV) の飛行制御が注目を集めている．特に，日本は最も多くUAVが利用されている国であり，その技術も先端的なものといえる．国産UAVの中では産業用無人ヘリコプタが最も多く用いられているが，これは他の航空機に比べて垂直離着陸や空中に停止することが可能であるなど優れた利点をヘリコプタが有しているために，国土面積が狭い日本における運用面からの要求を満たすからであろう．現在のところ，産業用無人ヘリコプタは主として農薬散布などの農業用途に用いられており，農作業の省力化に大きく貢献している [45]．飛行制御系の進歩により，その操縦は以前に比べて簡単になっているが，ある程度の訓練が必要であり，多くの場合は限られた熟練者が操縦を行なっている．また，無線による遠隔操縦であるので，操縦者は視覚によってヘリコプタの姿勢などの状態を認識し，その情報に基づいて制御を行なっているため，操縦者がヘリコプタから離れるにしたがって操縦は困難となる．よって，操縦者はヘリコプタからあまり距離をとることができなかった．操縦可能な範囲は熟練者で150mとされているが，100m程が大半の操縦者の限界であろう．これは無人ヘリコプタだけでなく，固定翼のUAVであっても同じであるために，これまでのUAVはその用途に制約が加わり，無人という特性を十分に活かすことができていなかったといえる．例えば，山岳地帯，特に火山など危険地域における観測活動，災害発生時における救難・救援支援活動などは，二次災害を避けるためにもUAVの活躍が期待される分野であるが，これまでは無人ヘリコプタを適用することは難しいと考えられていた．このような難点を克服し，無人ヘリコプタの特性を十分に活かすために，著者らとヤマハ発動機 (株) は自律飛行制御に関する研究を行なってきた [46]．その研究の成果として，無人ヘリコプタの自律制御による飛行実験に成功し，その結果に基づいて開発された自律型無人ヘリコプタは2000年の北海道有珠山の火山活動の観測や三宅島

の観測に用いられ，火山性噴出物による周辺地域の被害状況をはじめ，火口などを観測することに成功した [47,48]．これらの結果により，自律飛行制御を行うUAVは，危険地帯の観測活動をはじめとして，救難・救助活動，安全監視活動，防災活動などに非常に有効であることが示された．

しかし，現在の自律飛行制御系は簡単なPD制御系であったために，さらなる高性能化や高信頼度化が要求される．このような目的のために，6自由度の非線形フライトシミュレータがヤマハ発動機(株)により開発された．このフライトシミュレータを用いることで，高精度に無人ヘリコプタの運動を計算機上で再現できるが，ヘリコプタの非線形特性やパラメータなどが公開されていなかった．従来の制御系設計法では，制御対象に関する詳細な情報が不可欠であるため，制御系設計にこのようなフライトシミュレータを直接用いることは不可能である．このように，制御対象に関する情報が利用できず，シミュレータのみから制御系設計を行なうにためには，試行錯誤法による発見的な手法に頼らざるを得ない．しかし，この方法は効率が悪いだけでなく，ロバスト制御をはじめとして高性能な制御系の設計は実際に不可能といってよく，高度化あるいは高信頼化を目指した制御系設計にはフライトシミュレータを十分に活用することができなかった．

航空機の制御系の開発では，飛行試験を少なくすることによりリスクやコストの低減を図るために，飛行実験や風洞試験などのデータからフライトシミュレータを作成し，飛行制御系を構築・評価を行なうことが一般的である．このような航空機のフライトシミュレータの作成には，さまざまな分野の専門家の知識が必要であり，多くの知識が集約されているといえる．飛行制御系の設計にはこのような知識を十分に活用する必要であるため，その開発は難しく，必要なコストも大きくなっている．もし，フライトシミュレータのみから飛行制御系の開発が容易に行なうことが可能であれば，さまざまな分野の専門家に分散して存在する知識を有効に活用することが可能となる．これは航空機の制御系の開発のみでなく，一般の対象における制御系の構築の際にも当てはまることであり，このためには多くの専門家の高度な知識と膨大な飛行試験の結果に基づいて作られたシミュレータを制御系の設計のために十分に活用する方法が必要となる．

機密保持のために，制御対象に含まれる特性パラメータを全てを知ることは難しいことが多い．特に，航空機の空力微係数など詳細な情報を得ることは困難である．近年は，多くの機関で共同研究が行われている．しかし，共同研究のためには秘密事項に相当する情報まで共有する必要があるために，情報が外部へ漏洩するリスクを負うことになる．本章で対象とする無人ヘリコプタでも，その特性パラメータの多くは秘密とされており，フライトシミュレータにおいても，ヘリコプタのダイナミクスに関する部分は公開されていない．本章では，制御系の設計において対象の情報を用いず，シミュレータにニューラルネットワークの学習を組み込むことで自動的に制御系の構築を行なう方法を考察し，UAVの自律飛行制御系の構築に応用する．本研究で用いたフライトシミュレータでは，各時刻における速度，位置，姿勢などの機体の制御に必要な情報は学習アルゴリズムにおいても利用可能であるので，これらの情報のみから学習を行い，

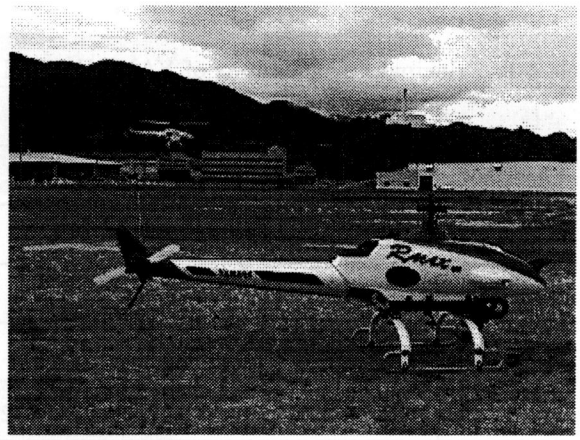


図 4.1: Unmanned Helicopter YAMAHA RMAX

制御系を構築する方法が必要である．よって，学習アルゴリズムをフライトシミュレータに組み込む際に，機体のパラメータなどの情報は用いることができないため，従来より提案されている学習方法はほとんど用いることができないが，提案する学習方法を組み込むことで，最適なフィードバック制御系をはじめとしてロバストな制御系が構築可能であることを示す．また，UAV の飛行制御において風は不確定な要素である．一般に，風は確率的に変動する外乱と考えられている．本章では，確率的な外乱に対してロバスト制御系を構築する方法をいくつか提案し，無人ヘリコプタの飛行制御シミュレーションを行なうことによりそれぞれの学習法の特性を明らかにする．本章において提案する学習法は，本研究で対象とした無人ヘリコプタだけでなく，一般の対象に対しても有効な方法である．

4.2 産業用無人ヘリコプタ

本研究で対象とする UAV はヤマハ発動機 (株) 製の産業用無人ヘリコプタエアロロボット RMAX [45] とする．図 4.1に RMAX の写真を示し，主な緒元を表 4.1にまとめる．表 4.1より，RMAX はコンパクトな機体でありながら，大きな実用ペイロードを持っていることが分かる．このためにさまざまな用途へ適用することが可能であると考えられ，NASA，カーネギーメロン大学，ジョージア工科大学など多くの研究機関において UAV の研究目的のために利用されている [9,49].

著者らはヤマハ発動機と共同で RMAX の自律飛行制御に関する研究を行ってきており，特に安全・防災活動へ応用を検討してきている．その結果として，1998 年に RMAX の自律飛行実験に成功した [46]．また，無人ヘリコプタの自律飛行の応用例として，北海道有珠山や三宅島における火山活動の観測に用いられた結果は，[47,48] に報告されている．

Table 4.1: Principal Specifications of YAMAHA RMAX

全長	3,630mm(ロータを含む)
メインロータ径	3,115mm
全高	1,080mm
全幅	720mm
重量	58kg
エンジン	水平対向 2 気筒 2 サイクル (排気量 246cc)
出力	21PS
実用ペイロード	30kg
飛行可能時間	1 時間

無人ヘリコプタの自律飛行が可能となったことにより，危険地帯の観測をはじめとして救難・救援活動など様々な活動に無人ヘリコプタを応用することができるようになったが，より高度なタスクを実行可能とするためにはさらなる制御系の高性能化や高信頼度化が要求される．このために，RMAX 専用の非線形フライトシミュレータが，ヤマハ発動機（株）において開発された．このフライトシミュレータにより，RMAX の運動を高精度にシミュレーションすることができる．また，Microsoft Windows 上で構築されているために，安価なパソコンでフライトのシミュレーションが可能である．このシミュレータで飛行シミュレーションを行なっている様子を図 4.2に示す．このフライトシミュレータには，C 言語で記述した飛行制御系を組み込むことが可能であり，実機と同じ飛行制御用のプログラムソースを用いることができるように，観測値などは全て実機と同一のフォーマットであり，制御や観測のサンプリング周期も実機と同じである．このため，フライトシミュレータで動作を確認されたソースコードは専用コンパイラを用いてコンパイルを行うことにより，速やかに RMAX で利用可能なコードに変換される．これにより，自律飛行制御系のシミュレーションによる検討から実機を用いる飛行試験への移行に必要な手続きが著しく簡略化され，飛行試験の回数の大幅な軽減も可能となった．しかし，このフライトシミュレータにおいて，RMAX の動特性に関する部分のプログラムは公開されていない．従来の制御系設計法では制御対象の情報を用いずに制御系を構築することは不可能であるため，制御系設計には十分に活用することができなかった．

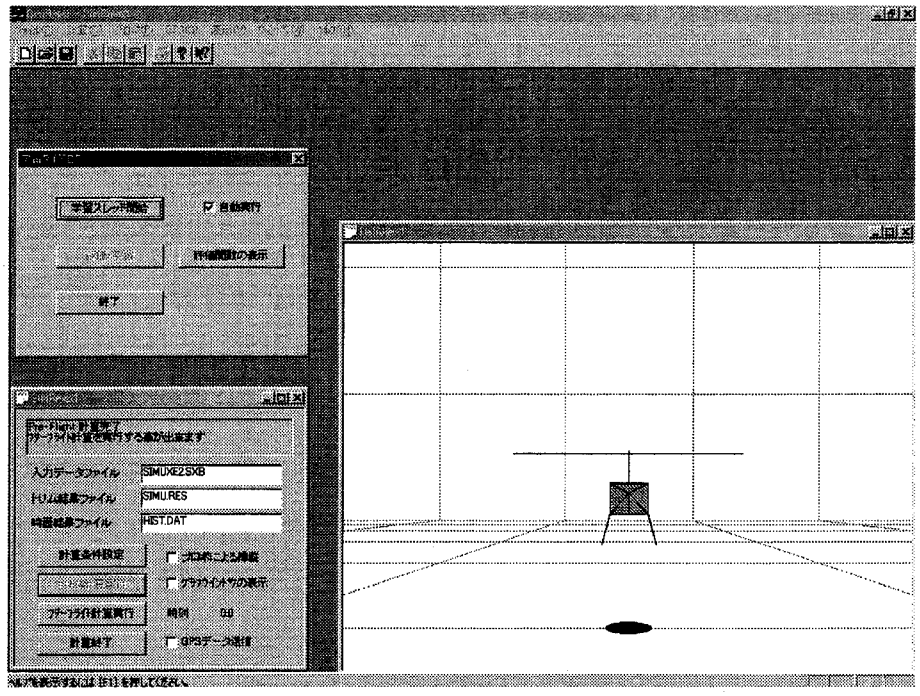


図 4.2: Flight simulator of YAMAHA RMAX

4.3 ニューラルネットワークの学習による制御系の構築

本節では，ニューラルネットワークの学習による制御系の構築について述べる．ここでニューラルネットワークの学習に関する内容は，制御対象として取り上げたRMAXに限らず一般的なものである．

4.2節で述べたように，動特性を記述する方程式などが公開されず，シミュレータのみが制御系設計に利用できるという状況では，線形制御系設計理論をはじめとする従来の制御系設計法は用いることができない．これは制御系設計に，制御対象の数学モデルが不可欠であるからである．フライトシミュレータの数値データより，無人ヘリコプタの数学モデルの作成を試みる方法も考えられるが，次のような理由で適切ではないと考える．

- フライトシミュレータにおいて無人ヘリコプタの数学的なモデルが数値計算のために用いられているが，そのモデルにはモデル化誤差が必ず含まれている．シミュレータの数値計算の結果から数学モデルを作成する際にも，モデル化誤差が残ることは避けられないので，得られた数学モデルが実際の機体の特性を正しく表しているかどうかは曖昧である．また，数学モデルを作る手間は小さくはないが，モデル化はシミュレータを作成する段階ですでに行なわれており，再びこれを行なうことは無駄な手続きである．

- 飛行実験などを繰り返した結果や専門家の知識に基づき，作成されたフライトシミュレータを十分に活用できない．

制御対象の情報を全く用いることなく，シミュレータなどから制御系を設計することができれば，4.1節で述べたように制御系設計の効率を向上することが可能なだけでなく，情報の漏洩のリスクを軽減することもできる．このために，制御対象の情報を一切用いずにシミュレータのみから制御系設計を行なう方法について考える．この場合，従来の制御系設計法を適用することができないが，制御系のゲインを試行錯誤により決定する方法が考えられるが，このためには多大な時間と労力を必要である．本研究では制御系の設計にニューラルネットワークの学習を応用することにより，このような問題点の解決を行う．

従来から提案されているニューラルネットワークの学習を利用した制御系の構築方法は大きく二つに分けられる．一つはオンライン学習の応用であり，各時刻において制御対象の特性の学習を行う方法であって，この方法は適応制御の範疇に含まれる．他方はオフライン学習の応用であり，ある評価関数を基準として学習を繰り返し行うことにより制御系を構築する方法である．本研究ではシミュレータにオフライン学習を組み込むことにより制御系の構築を行う方法を考察する．これは，オンライン学習は任意の制御目的に対応できるというわけではないことと，オフライン学習により構築した制御系の方が一般に高性能であるからである．しかし，オンライン学習が非常に有用である制御系もある．特に，再構築可能な飛行制御系 [50,51] では，飛行中に生じた障害などを検知し，制御系の再構築を飛行中に動的に行うことにより安全な飛行制御を行うことを目的としているが，ニューラルネットワークのオンライン学習が非常に役に立つ制御系といえる．このような制御系はオフライン学習により得られた制御系の補助として用いられるものである．

第2章において提案した勾配法に基づく学習アルゴリズムでは評価関数の勾配を計算する必要がある．このためには制御対象のヤコビアンが不可欠となる．それゆえ，勾配法に基づく学習アルゴリズムにより制御系の構築を行うには，制御対象の正確なモデル，あるいは近似モデルが必要であり，学習アルゴリズム自体にに制御対象の情報を組み込む必要がある．このために，学習アルゴリズムを構築する際に制御対象に関する情報が不可欠であり，シミュレータのみを用いて制御系を学習させることはできない．これに対して，Powell の共役方向法に基づく学習アルゴリズムは，微分不可能な評価関数や活性化関数を持つ場合に適用するために導かれたものであるが，探索点での評価関数の値のみに基づいて学習が行なわれるため，学習アルゴリズムに制御対象の情報は不要である．このために，この学習アルゴリズムをシミュレータなどに組み込むことで，制御対象の情報を全く用いずに学習を行うことが可能である．一般のシミュレータに，この学習方法を組み込むことができるための条件は以下の通りである．

- 各サンプル時刻における制御対象の状態を用いることができ，制御系の評価関数が計算できること．
- 学習ルーチンから繰り返し計算など計算の制御することができること．

第一の条件は，遊技や運転訓練を目的として作成されたシミュレータでなく，制御系の設計や評価を行なう目的で作成されたシミュレータならば満たされているであろう．特に，フィードバック制御を行なう場合は，対象の状態は制御ルーチンに伝えられるため，評価関数の計算は必ず可能となる．また，この条件が満たされるように変更しても，対象に関する情報を漏洩することにはならないとため，変更は容易であると思われる．第二の条件は，学習の自動化のために必要な条件であって必須ではないが，多くのシミュレータでは計算の制御を行うことにより，繰り返し計算を行なうことは可能であろう．このように，ニューラルネットワークの学習による制御系の設計をシミュレータに組み込むことは，一般に容易であると考えられる．実際に，RMAX のフライトシミュレータにおいて，上記の条件がいずれも満たされていたため，学習アルゴリズムをシミュレータに組み込むことは容易であった．

4.4 自律飛行制御系の構築

UAV の自律飛行制御系を開発するにはハードウェア，ソフトウェア両方の開発が必要であるが，本研究はソフトウェアの開発に関わるものである．UAV の自律飛行のためのソフトウェアには飛行制御系の構築だけでなく，航法システムなども含まれる．このために，自律飛行制御用ソフトウェアの開発には，図 4.3 のような階層的なアプローチが適すると考えられる．上位層には主としてアプリケーションが属する．コマンドシステムなどの操縦インタフェースや画像認識などによる状況判断，飛行経路の計画，航法システムなどは上位層に属する．中位層は，主として自律飛行制御系の信頼性に関与し，制御モードの切換や再構築，変動に対する適応などがこれに属する．また，飛行データによる異常診断も中位層に属する．下位層には飛行制御が属し，自律飛行制御系の最も基本となる層である．このように階層的なアプローチによって，自律飛行制御系の開発が分散的に効率よく行なうことができる．図 4.3に従うと，ニューラルネットワークによる制御系の構築は，中位，あるいは下位層に属する．

本研究では階層構造型ニューラルネットワークを制御系として用いる．ニューラルネットワークを利用することにより，さまざまな制御系の構築が可能となる．例えば，航空機分野において最も多く用いられている制御の一つにゲインスケジューリング制御系があるが，この設計を行うためには非常に多くの飛行試験や労力が必要である．しかし，図 4.4に示したブロック図のようにニューラルネットワークを配置し，評価関数 (4.1) に基づいて学習を行うことにより，容易に構築することができる．

$$J = \sum_{t=0}^{T} L(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{4.1}$$

ここで，L は非負の関数であり，$\boldsymbol{x}(t)$，$\boldsymbol{u}(t)$ はそれぞれ時刻 t におけるヘリコプタの状態変数，制御入力である．このようなゲインスケジューリング制御系の構築には中間ユニットの活性則としてシグモイド関数を

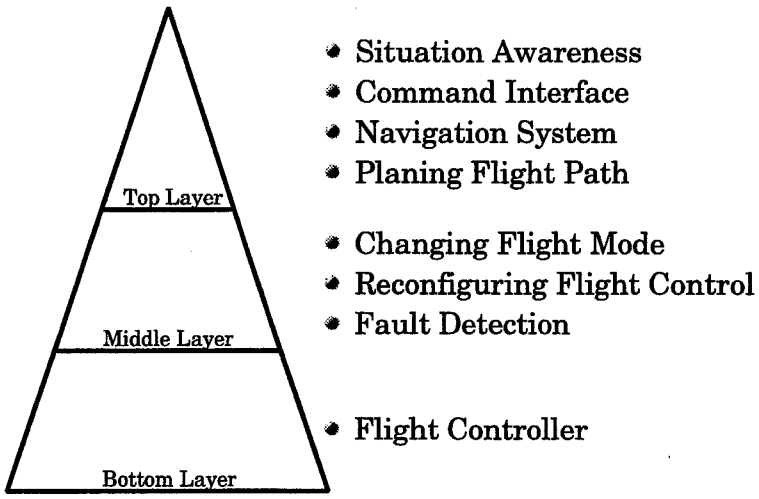


図 4.3: Hierarcy structure of autonomous flight control systems

使うネットワークを用いるよりもRBFニューラルネットワークの一種であるガウシアンニューラルネットワークを用いる方が適している [52]. シグモイド型の活性では入力ユニットの活性値I_jとして結合度w_{ij}である中間ユニットの活性値H_iは，しきい値θ_iおよびシグモイド関数Fを用いて

$$H_i = F(\sum_j w_{ij} I_j + \theta_i) \tag{4.2}$$

と表されるのに対して，ガウス型の活性則ではσ_{ij}およびθ_{ij}を用いて以下のように表される.

$$H_i = \exp\left(-\sum_j \frac{(I_j - \theta_{ij})^2}{\sigma_{ij}^2}\right) \tag{4.3}$$

このように，シグモイド型の活性では重み付け入力和によって中間ユニットが活性するのに対して，ガウス型の活性則では入力データのパターンにより中間ユニットは活性化する．ゲインスケジューリング制御は対象の動作点により制御系のゲインの変更を行うものであるのでシグモイド型の活性則よりガウス型の活性則の法が適している．また，シグモイド型の活性則は大域的であるが，ガウス型の活性則は局所的であるので，中間ユニットと各ゲインとの対応が理解しやすくいだけでなく，学習後のゲインの微調整が容易となるなどという利点があるため，ゲインスケジューリング制御系の構築に適していると考えられる．しかし，一般にRBF型の活性則を持つニューラルネットワークはシグモイド型活性則のものに比べると近似能力が劣るので，図4.5であらわされるような非線形フィードバック制御系の構築には，シグモイド型の活性則をもつニューラルネットワークを用いる方がよい．このとき (4.1) に基づいて学習することにより，ニューラルネットワークは非線形最適フィードバック制御系となる.

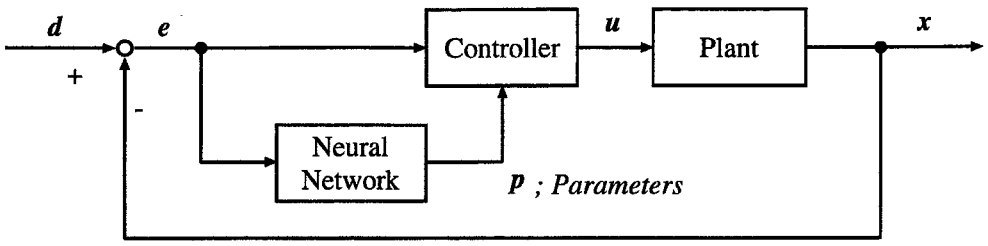


図 4.4: Gain scheduling controller

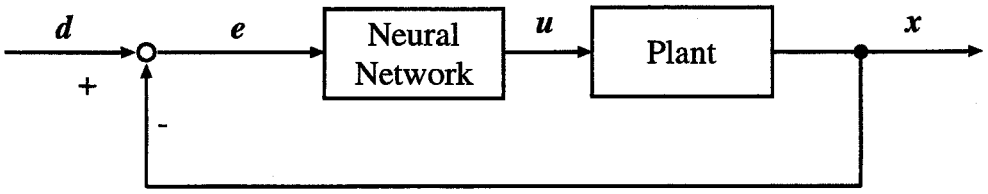


図 4.5: Nonlinear feedback controller

航空機やヘリコプタのダイナミクスは，一般に (4.4) のように非線形方程式で表される．

$$\ddot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{u}) \tag{4.4}$$

(4.4) の左辺を$\boldsymbol{U}$とおき，$\boldsymbol{U}$を疑似入力と呼ぶ．

$$\boldsymbol{U} = \boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{u}) \tag{4.5}$$

$\boldsymbol{f}$の逆が存在し，(4.6) を満足する制御入力$\boldsymbol{u}$が存在するとき，

$$\boldsymbol{u} = \boldsymbol{f}^{-1}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{U}) \tag{4.6}$$

疑似入力$\boldsymbol{U}$と出力$\boldsymbol{y}$との関係は線形であるので，非線形システム (4.4) は線形非干渉システム (4.7) に変換される．

$$\ddot{\boldsymbol{y}} = \boldsymbol{U} \tag{4.7}$$

(4.6) ような入力の変換を逆ダイナミクス変換といい，航空機やヘリコプタの制御に応用する研究が数多くなされている [28,29]．疑似入力$\boldsymbol{U}$を (4.8) のような線形 PD 制御入力と定めると，

$$U = -K_p(y - d) - K_d\dot{y} \tag{4.8}$$

$$\ddot{y} = -K_p(y - d) - K_d\dot{y} \tag{4.9}$$

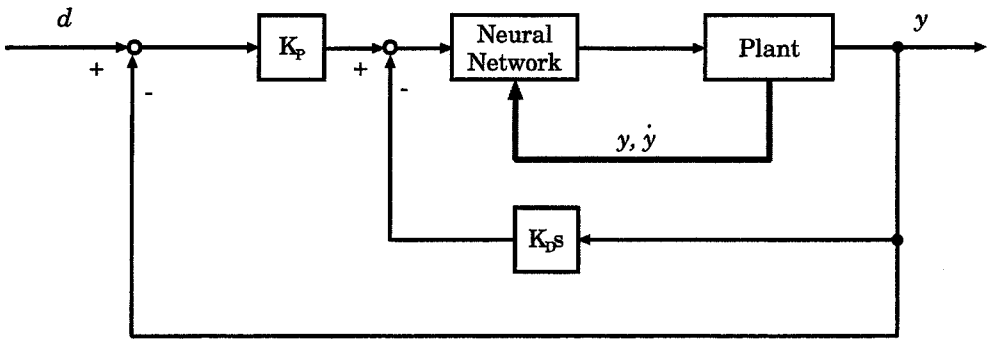


図 4.6: Controller for linearization by a neural network

非線形なダイナミクス (4.4) は，逆ダイナミクス変換により (4.9) のように線形のダイナミクスとなる．$\boldsymbol{d}$は目標出力であり，ゲイン係数 K_p，K_dは閉ループ特性 (4.9) が設計仕様を満たすように選ぶ．明らかなように，逆ダイナミクス変換を用いる制御系の設計には，制御対象の詳細な知識が必要である．制御対象の詳細な知識が存在しているならば，その知識に基づいて逆ダイナミクス変換を求めればよいので，ニューラルネットワークの学習を用いる必要はない．しかし，対象のダイナミクスに対する情報が利用できず，シミュレータのみを用いて制御系の構築を行う場合には，ニューラルネットワークの学習により逆ダイナミクス変換を得なければならない．逆ダイナミクス変換にニューラルネットワークを用いた制御系のブロック図を図 4.6に示す．図 4.6のブロック図では逆ダイナミクス変換の学習を行うことができず，学習は図 4.7に示したブロック図のシステムを用いる．ニューラルネットワークの学習の目的は線形化誤差の最小化であるので，学習の評価関数として (4.10) を用いる．

$$J = \sum_{t=0}^{T} e^2(t) \tag{4.10}$$

自律飛行制御を行なう RMAX には，GPS アンテナ (ノバテル社製 RT-20) が装備されており，高精度の位置・速度計測を行なうことができる．また，機体には慣性ユニットも装備されており，加速度や角速度，姿勢角などを測定することができる．本研究では，制御をはじめとして学習にも全て測定可能な状態のみを用いる．また，サンプリング周期は実機と同じ 20msec とする．ヘリコプタの制御入力はエレベータ操舵指令δ_e，エルロン操舵指令δ_a，コレクティブピッチ操舵指令δ_c，ラダー操舵指令δ_rの 4 つである．一般の航空機と同様に，構造的な理由からそれぞれの操舵角には制約がある．それぞれの入力利用可能範囲はフライトシミュレータに組み込まれており，ある大きさ以上の操舵量を入力しても利用可能範囲外であれば飽和により限界量を越えないように自動的に修正される．エレベータ，エルロンおよびラダーは主としてそれぞれ機体のピッチ角，ロール角およびヨー (方位) 角の制御に使われ，コレクティブピッチ操舵指令は高度

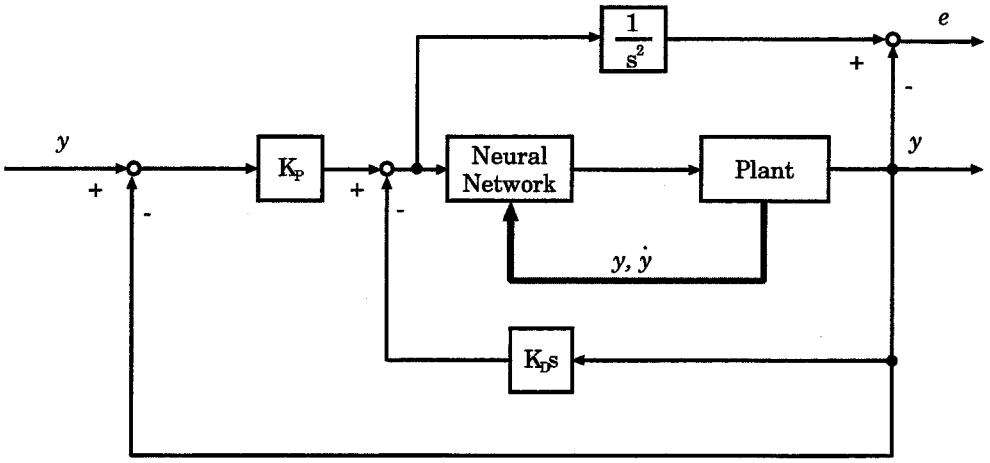


図 4.7: Block diagram for training a neural network

制御に使われる．ただし，実際には機体の姿勢角など状態量は独立ではなく互いに干渉する．

観測活動などにおいてはUAVの位置制御が重要となるが，自律移動においては速度制御も重要となる．図4.3に示した自律飛行制御系では，上位層における制御目的の変更により，中位層において制御系のモード切換が行なわれる．ニューラルネットワークの学習によると，位置制御系または速度制御系のいずれの構築を行うことも可能である．

4.5 自律飛行制御例

4.5.1 水平方向の位置制御

本節では，水平面内の位置制御系を構築した例を示す．機体の姿勢角などは独立ではなく互いの運動は干渉するが，ここでは簡単のために各機体軸方向および方位角制御は独立であると仮定する．また，高度制御，方位角制御は適切に定めた線形フィードバック制御系であるとし，それぞれ高度，方位角は一定に保たれるように制御している．高度制御，方位角制御が適切に定められていれば，運動の干渉が制御系の学習に大きな影響を与えることはなかった．

まず，エレベータ制御系を図4.5のような非線形フィードバック制御系を用いて構築する．このときの学習では，次のような評価関数を設定し学習を行なった．

$$J = \sum_{t=0}^{T} (x(t) - d(t))^2 + v_x^2(t) + 10\theta^2(t) + 10^3 q^2(t) \tag{4.11}$$

ここで $x(t)$，$v_x(t)$ はヘリコプタの機種方向の位置および速度，$\theta(t)$，$q(t)$ はピッチ角度とピッチ角速度である．Tは終端時刻であり，ここでは60秒と設定し学習を行った．また，$d(t)$ は目標位置であり，次式の

ように定めて学習を行なった．

$$d(t) = \begin{cases} 0.0 & (0 \leq t < 0.5) \\ 3.0 & (0.5 \leq t < 30) \\ 0.0 & (t \geq 30) \end{cases} \tag{4.12}$$

ネットワークの入力は$x(t)$,$v_x(t)$，$\theta(t)$，$q(t)$の4つとし，出力はエレベータ操舵角$\delta_e(t)$である．また，用いたニューラルネットワークの中間ユニット数は7個とし，シグモイド型の活性則を用いた．

本研究で用いたフライトシミュレータでは，あらかじめ決められた飛行範囲を越えたり，ロータの推力が収束しなかったなど明らかに飛行制御が失敗した時点で飛行シミュレーションが強制終了された．このために，T秒までに制御が失敗したときには，次のような評価関数でを用いることとした．

$$J = C + \frac{T}{\tau} \tag{4.13}$$

(4.13)における第一項は飛行制御失敗のペナルティを表し，十分に大きな定数である．第二項にあらわれるτは制御が失敗し計算が中断された時刻である．第二項により，飛行制御が失敗している間は制御による飛行時間を伸ばすように学習が進むようになる．当初は，制御が失敗する時点まで計算した評価関数(4.11)の値に一定の大きさのペナルティを加えていた．しかし，このようにすると短い時間で計算が中断させてしまうことにより評価関数が小さくなることが多いため，可能な限り短い時間で制御を失敗するように学習が進むことがあった．このため，学習がうまく進まないことが多かったが，制御が失敗のしたときの評価を(4.13)とすることにより，学習が進まないまま終了するといったことはほとんどなくなった．しかし，学習に不連続な評価関数(4.11)，(4.13)を用いることとなる．また，そもそも(4.13)は微分不可能であり，勾配に基づく学習アルゴリズムではこのような評価関数により学習を行なうことはできない．このような評価関数を用いて学習を行なうことが可能となることは，Powellの共役方向法に基づく学習法の利点の一つである．学習後のネットワークによるヘリコプタの制御結果と総当たり法により発見した最適なPD制御系による制御結果を図4.8と図4.9に示す．図4.8は機種方向の位置の応答を，図4.9はそれぞれのときのピッチ角の変化を示している．PD制御における評価関数(4.11)の値は4.739×10^3であったのに対して，学習により得られたニューラルネットワークによる制御では3.273×10^3であり，ニューラルネットワークによる大幅な性能の改善が確認できた．

次に，エルロン制御系の学習の例を示す．ここでは，図4.4のようなゲインスケジューリング制御系をニューラルネットワークの学習を用いて構築する．$y(t)$，$v_y(t)$，$\phi(t)$，$p(t)$をそれぞれヘリコプタの左右方向の位置，速度，ロール角度，ロール角速度とする．y方向の目標位置をdとすると，制御入力δ_aは次のように表される．

$$\delta_a(t) = P_y(t)\{d(t) - y(t)\} - P_v(t)v_y(t) - P_\phi(t)\phi(t) - P_p(t)p(t) \tag{4.14}$$

ここで，ニューラルネットワークへの入力を $d(t)-y(t)$，$v_y(t)$，$\phi(t)$，$p(t)$，出力を $P_y(t)$，$P_{v_y}(t)$，$P_\phi(t)$，$P_p(t)$ とする．また，学習に用いる評価関数は (4.15) のように設定した．

$$J=\sum_{t=0}^{T}10\left(y(t)-d(t)\right)^2+v_y^2(t)+\phi^2(t)+p^2(t) \tag{4.15}$$

目標位置 d を次のように設定したときの，学習後のニューラルネットワークによる制御結果を図 4.10，4.11 に示す．

$$d(t)=\begin{cases}0.0 & (0\le t<0.5)\\ 2.0 & (0.5\le t<30)\\ 0.0 & (t\ge 30)\end{cases} \tag{4.16}$$

さらに，エレベータ制御系にもゲインスケジューリング制御系を学習させたニューラルネットワークを用い，(4.17) に示された軌道を目標値として，軌道追従制御を行なった結果を図 4.12に示す．この図から，ニューラルネットワークを用いた制御系により，大きな誤差がなく目標軌道に追従する飛行を行なうことが可能であることが分かる．

$$\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}2\cos(N\sigma)\cos(\sigma)-2\\ 2\cos(N\sigma)\sin(\sigma)\end{bmatrix}\quad \sigma\in[0:2\pi]\ \ N=1,2,\cdots,5 \tag{4.17}$$

4.5.2 方位角制御

本節では，RBF ニューラルネットワークによるゲインスケジューリング制御系の学習を方位角制御に対して適用した例を示す．中間層のユニット数を 3 個とし，ガウス型の活性則を持つとする．ネットワークへの入力は$\psi_d-\psi$と rとした．ここで，ψを方位角，ψ_dを目標方位角 rは方位角速度である．ネットワークの出力はゲインスケジューリング制御器のゲイン係数 P_ψ，P_rであり，制御入力であるラダー操舵量δ_rは次のように表される．

$$\delta_r(t)=P_\psi(t)(d_\psi(t)-\psi(t))+P_r(t)r(t) \tag{4.18}$$

実際には，δ_rの大きさも他の操舵量と同様にその利用には上限と下限がある．自律型 UAV の方位角制御において方位角は目標方位角に速やかに追従し，その角度を安定に保つことが必要となる．このために，目標方位角を次のように変化させて制御系の学習を行なった．学習には (4.20) の評価関数を用いた．

$$\psi_d(t)=\begin{cases}0.0 & (0\le t<0.5)\\ \pi/3 & (0.5\le t<10)\\ 0.0 & (t\ge 10)\end{cases} \tag{4.19}$$

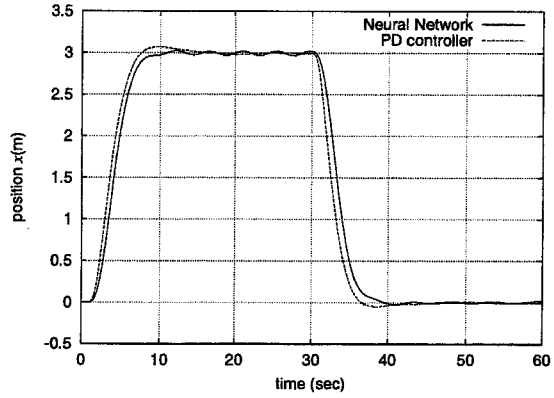


図 4.8: Responses to desired position $d(t)$ in forward control

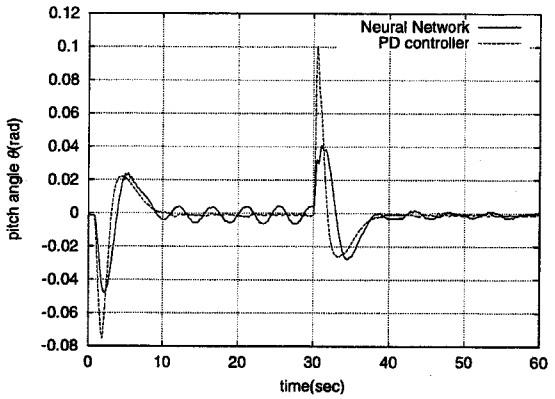


図 4.9: Transitions of pitch angle θ in forward flight

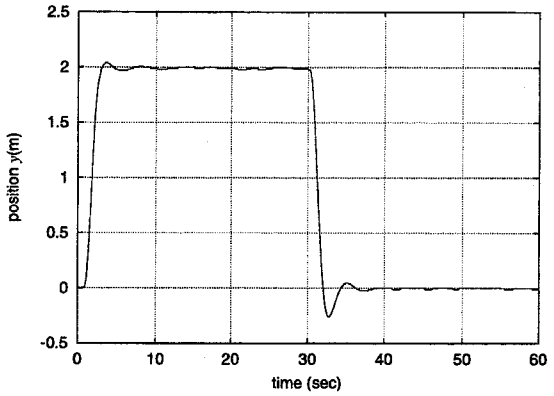


図 4.10: Responses to desired position $d(t)$ in lateral control

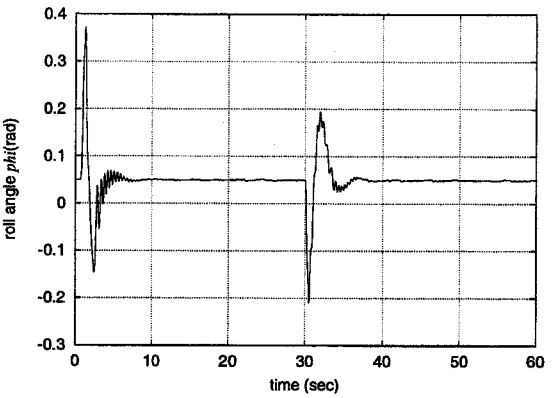
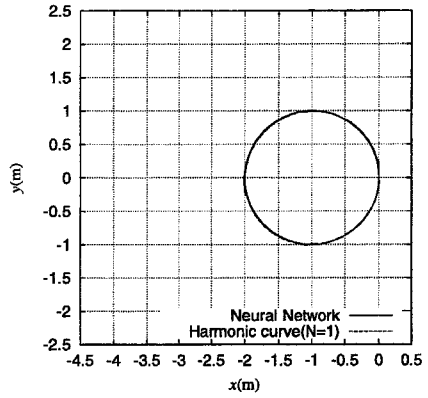
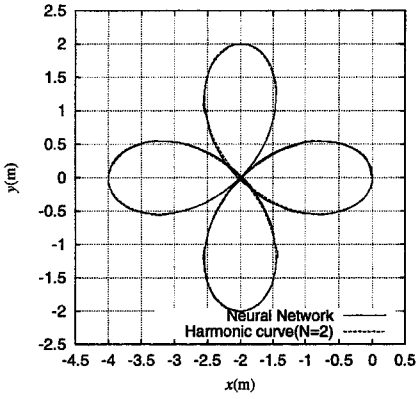


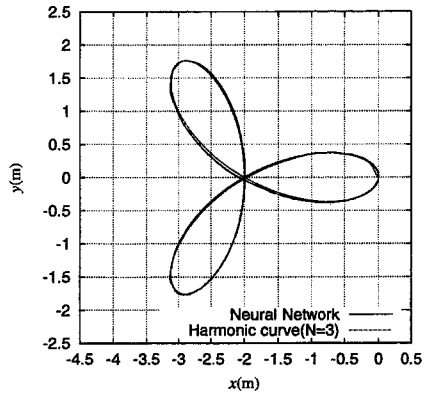
図 4.11: Transitions of roll angle θ in lateral flight



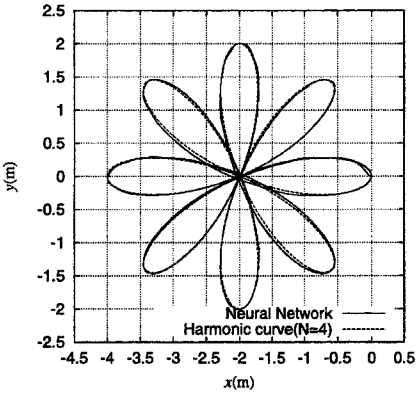
(a) $N = 1$



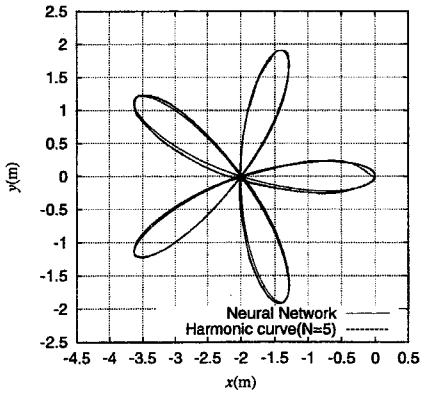
(b) $N = 2$



(c) $N = 3$



(d) $N = 3$



(e) $N = 5$

図 4.12: Tracking to harmonic curves

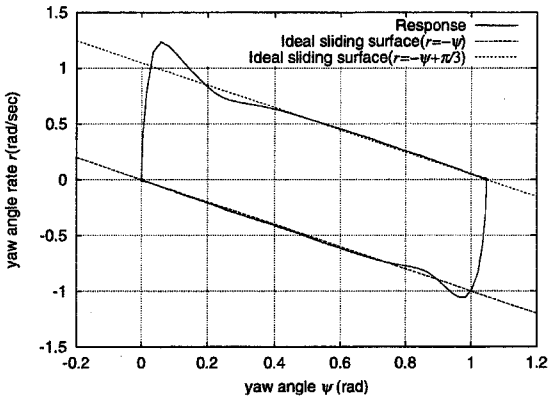


図 4.13: Response of the azimuth controlled by a trained neural network

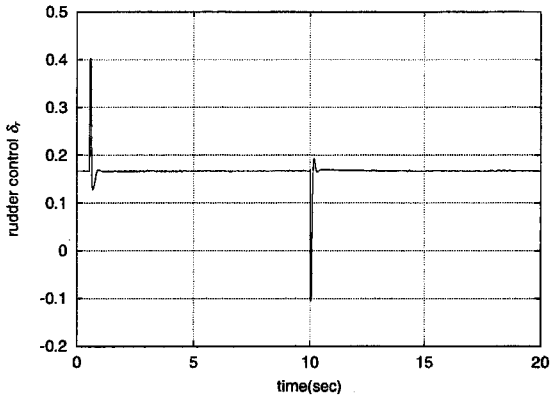


図 4.14: Control of rudder

$$J = \sum_{t=0}^{T} (d_\psi - \psi)^2 + r^2(t) \tag{4.20}$$

方位角ψと方位角速度rを図 4.13に示す．ラダーサーボが十分早く反応するなどの仮定を設けると，RMAX の方位角のダイナミクスは 2 次の運動方程式で記述できると近似することが可能である．第 3章において示したように，(4.20) のように制御入力を陽に含まない評価関数を用いる場合には，学習後の制御系はスライディングモード制御系となることが予想される．実際に，図 4.13をみると，状態はほぼ完全に滑り面に拘束されていることが分かる．また，このときの制御入力δ_rを図 4.14に示す．この図によると，制御入力δ_rは飽和による制約を受けていることが分かるが，スライディングモード状態においても入力にはチャタリングが生じておらず，良好な結果となっている．また，方位角が目標方位角のときであっても，δ_rは 0 ではなく，0.167 という値が付加されているが，これはメインロータのトルクの反作用による回転を打ち消すために必要となるテールロータの平衡推力を発生させるためのものである．さらに，図 4.14によるとこの平衡入力からのずれの大きさの最大値は回転の方向によって異なっていることが分かる．このように制御入力の大きさからも，RMAX の方位角制御は非線形性を有することは明らかであり，制御系の設計が難しい対象である．

4.5.3 高度制御

本節では非線形逆ダイナミクス変換を用いた制御系の学習を行なうことにより高度制御系を設計した例を示す．高度制御系のブロック図は図 4.6に示されるものと同一であり，ここでは逆ダイナミクスの学習が完全完了したときの閉ループ系の周波数特性やステップ関数への追従性から $K_p = 1.0$, $K_d = 2.0$ と選んだ．また，ここでもコレクティブ制御以外の制御系は適切に定めた PD 制御系であるとしている．学習が終了したニューラルネットワークにより制御された RMAX の応答と理想的な線形モデルの応答を図 4.15に示す．

オフライン学習により得られた逆ダイナミクス変換を学習したニューラルネットワークと制御系のオンライン学習を行なうニューラルネットワークは併用することが容易であり，併用することにより耐故障性のある制御系に発展させることが可能である．オンライン学習と併用することの有効性を示すために，電動ヘリコプタを用いた方位角制御実験を行なった結果を示す．ここでは，オンライン学習法としてはCaliseらの提案する方法 [9,10] を用いた．図4.17は実験開始5秒後に一定速度の横風を加えたときの実験結果である．明らかなように，オンライン学習が無い場合には，風の影響により方位角を0に保つことができなかった．これは，制御系を設計したときに用いた数学モデルでは横風の影響によるラダー操舵量の平衡点のずれが考慮されていないからである．これに対してオンライン学習が存在すると，風の影響を打ち消すようにラダーの平衡点が移動するように学習が進み，風が存在しても方位角を0に保つことができた．またラダーの効果が半減してしまうような故障の発生を模擬するために，ラダー制御用の電圧が通常の50%しか出力されないようにして実験を行なった結果を，図4.17に示す．これらの結果から明らかなように，オンライン学習を併用することにより耐故障性のある制御系を構築することが可能であることが分かる．しかし，オンライン学習のみでは効率よく飛行制御系を構築できないので，はじめにオフライン学習により制御系を構築する必要がある．オフライン学習による制御系の設計とオンライン学習による制御系の適応を併用することによって，耐故障性のある制御系など信頼性に優れた制御系を構築することが可能であると考えられる．

4.6 確率的な不確かさに対するロバスト制御系の学習

4.6.1 確率的な不確かさに対するロバスト性

4.4節において述べた自律飛行制御系の構築法は，本論文の2章で述べた最適フィードバック制御系の応用である．しかし，この方法で学習を行なったニューラルネットワークではロバスト性が問題になることは3章で指摘した通りである．定量的なロバスト制御系を学習する方法は3章において詳しく述べたが，そこで提案された方法はパラメータの不確さや非構造的な不確さなど確定的な不確さが対象となっていた．しかし，UAVの飛行制御においては最も代表的な不確かさは風である．このために，自律飛行制御系の学習に用いたフライトシミュレータでは次に示す3種類の風が用意されており，風に対するRMAXの応答をシミュレーションすることが可能であった．

- 孤立突風

  ある時刻に，定められた大きさでインパルス状に吹く仮想的な風

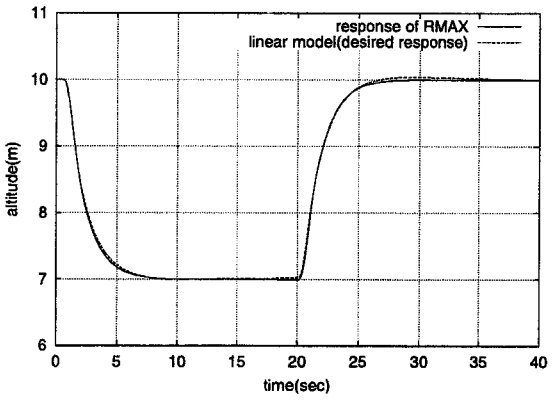


図 4.15: Altitude response controlled by a trained neural network

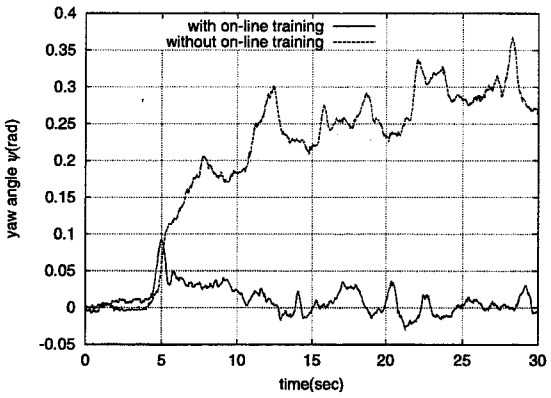


図 4.16: The effect of on-line training(side wind began to blow from 5sec)

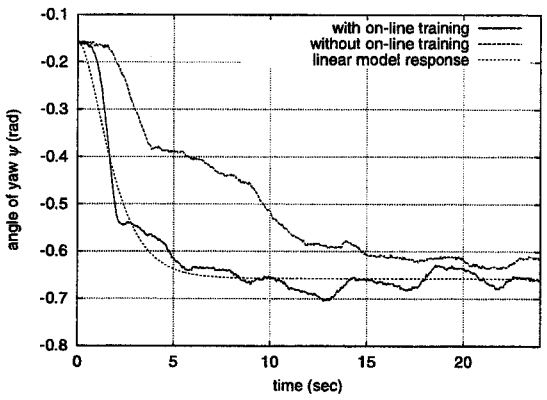


図 4.17: The effect of on-line training(efficiency of the rudder is decreased to 50%)

- ランダム風

  持続的にランダムな大きさで吹く仮想的な風

- 再現風

  実測データを用いて再現された風

火山など山岳地帯では乱気流の発生が考えられるため，無人ヘリコプタの救難・救助活動では飛行制御系の風に対するロバスト性が重要である．風の大きさや向きは不規則に変動することが知られており，飛行制御系を構築する際には白色雑音あるいは，先験的に知られているスペクトル分布をもつ有色雑音として近似することが多い．このため，シミュレーターに用意されていた風の中では，孤立突風は現実の風と近い状況を再現するとはいえない．また，限られた実測値による再現風では，測定データに偏りがあり可能性がある．必ず同じ風が吹くという状況下では特性のよい制御系が学習することが可能であるが，実際にそのような状況はあり得ない．再現風を用いて学習を行なうと，特殊な風に対するロバスト性が確保されるだけであり，再現風以外の風のパターンに対するロバスト性は全く保証されないので，再現風を用いて制御系設計や評価を行うことは危険である．このために，ロバスト制御系を構築する際は用意されている風の中からランダム風を用いるのが妥当である．しかし，すでに提案したロバスト制御系の学習法は確定的な不確さを取り扱うことが可能であるが，確率的な不確さを扱うことができない．そこで，風に対してロバストな制御系を構築するためには，確率的な不確さに対するロバスト制御系の学習法が必要となる．本節では，確率的な不確さに対するロバスト制御系の学習法をいくつか提案し，数値例によりその特性や有効性を確認する．

4.6.2 J_γを用いる学習

制御対象に確率的な不確かさが加わることにより，制御結果から算出した評価関数も確率的に変動する．このために，2章で提案した学習法では正しく学習をおこなうことができない．確率的な不確かさを計算機に記憶させることにより，その再現を行なうことは可能である．しかし，ある特殊な不確かさのパターンにのみを考慮した学習となり，その他の確率的な不確かさのパターン対してもロバスト性を持つとはいえない．このために，制御系の評価としては確率的に変化する評価関数の統計量を用いねばならない．よって，学習も同様に統計量に基づいて行なう必要がある．代表的な統計量としては平均値や分散などがある．確率的な不確かさにより，評価関数はその平均値を中心としてにある大きさの偏差を持って分布する．このような偏差が大きいことは，制御系が確率的な不確かさにより大きく乱されることを意味するので，好ましくないといえる．仮に，評価関数の確率的な変動を無くすことができとすると，完全に確率的な不確かさの影響を取り除くことができることになるが，制御系によりある程度までロバスト性を高めることができても，風など確率的な不確かさの影響を完全に取り除くことは実際には不可能であろう．また，一般にロ

バスト性の向上により制御性能は低下してしまうため，仕様によって定められた適切な大きさのロバスト性を持つ制御系の設計を行わなければならない．このために，スカラーγをパラメータとして含む評価関数(4.21)を用いて学習することを考える [53].

$$J_\gamma = \frac{1}{2\gamma}\log\left(E\left[\exp(2\gamma J)\right]\right) \tag{4.21}$$

ここで，$E[J]$はJの期待値を示す．Jは制御結果の標本から求めた評価関数である．γに関するTaylor展開により，(4.21)は(4.22)のよう展開できる．

$$J_\gamma = E\left[J\right] + \gamma Var\left[J\right] + O(\gamma^2) \tag{4.22}$$

ここで$Var[J]$はJの分散を示す．(4.22)より明らかなように，この学習法はγによって3つの場合に分類することができる．

1. $\gamma = 0$　ただし，この場合はJ_γは$E[J]$とする．評価関数Jの平均のみを考慮する学習方法.
2. $\gamma > 0$　評価関数Jの平均値とその分散を考慮する学習法．分散が大きくなることを防ぐことにより，評価関数が期待値から大きく隔たる値が得られることを防ぐ学習方法.
3. $\gamma < 0$　評価関数Jの平均値とその分散を考慮する学習法．分散が小さくなることを防ぐことにより，評価関数が期待値から大きく隔たる値が得られることを許容する学習方法.

評価関数Jの平均が小さくてもその分散が大きい場合，その制御系は風による影響を強く受けることを意味するので制御系として好ましくない．このために，実際にロバスト制御系の学習を行なう際に用いるのは$\gamma \geq 0$に限られる．また，この学習はH_∞外乱除去問題を解くことと同じであり，γが大きいほど確率的な不確かさに対してロバストとなり，外乱除去性が高いといえる．しかし，(4.21)の計算のためには指数関数の計算を行う必要がある．このため，計算を行う際に十分に注意をしなければ，計算結果が発散したり数値誤差が大きくなりやすい．このような計算の困難さを避けるために，

$$J_\gamma = E\left[J\right] + \gamma Var\left[J\right] \tag{4.23}$$

ように評価関数を設定して学習を行なうことも考えられる．複数の目的関数を最小化する多目的最適化問題において，ベクトル値であった評価関数を重み係数を用いてスカラーの評価関数に変換する方法を重み係数法という．J_γを用いる学習は，Jの平均と分散を共に最小にする多目的最適制御問題をスカラーγを重み係数として用いた重み係数法により変換した単一目的の最適制御問題と等しいことが分かる．一般に多目的最適化問題には多数のパレート最適解が存在することが知られている．よって，異なるγである(4.23)を用いて学習を行なったときの結果は，多目的最適化問題のパレート最適解となる．

4.6.3 J_vを用いる学習

J_γを用いる学習では，Jの平均と分散の両方が考慮されることが分かった．また，γを0とすることによりJの平均のみが学習に考慮されるので，Jの平均のみを考慮する学習はJ_γを用いる学習に含まれる．これに対して，(4.23)における$\gamma \to \infty$の極限である評価関数(4.24)により学習を行うことも考えられる．

$$J_v = Var[J] \tag{4.24}$$

しかし，J_vを最小化する学習により得られる制御器は一般に安定であるとはいえない．例えば，本研究で利用したフライトシミュレータによる数値計算では，飛行失敗のときには(4.13)のような評価関数に基づいて制御系の評価を行なうために，すべての試行で早い時刻に制御失敗するときJの分散は小さなものとなる．よって，(4.6.4)をそのまま制御系の学習に用いると，不安定な飛行制御系が得られるために十分ではない．安定な制御系の中でJの分散が最小となるものを学習するために，次の等式拘束条件あるいは不等式拘束条件を付加してJ_vを最小化するように学習する方法が考えられる．

$$E[J] - E_v = 0 \tag{4.25}$$

$$E[J] - E_v \leq 0 \tag{4.26}$$

ここで，E_vは設計パラメータであり，設計者が選ぶことができるパラメータである．すでに述べたようにJの分散が小さいということは確率的な不確かさの影響が小さく，制御系が高いロバスト性を持っていることを意味する．これに対して，Jの平均が小さいことは制御系が高い制御性能を持つをこと意味する．すなわち，分散の最小化と平均の最小化は一般に相反する要求であり，一般に多数のパレート最適解が存在する．J_γを用いる学習法とJ_vを用いる学習法は，Jの平均と分散の多目的最適化問題のパレート最適解の一つを求めるという観点からは同じものであるが，その特性は異なるものとなる．

本学習法で用いる拘束条件にとしては，等式拘束条件(4.25)と不等式拘束条件(4.26)の二つが考えられる．等式拘束条件(4.25)を用いると選択すると，$E[J] = E_v$が満足されるように学習が進む．しかし，図4.19に示した線形制御系によるJの平均と分散の関係から，Jの平均がある値までは平均が同じである制御系の分散の最小値が小さくなっていくことが分かる．しかし，Jの平均がある値を越えると，逆に平均が同じである制御系における分散の最小値が大きくなっていくことも分かる．制御性能とロバスト性がともに優れた制御系を学習することが目的であるので，通常は不等式拘束条件(4.26)を用いるほうがよいと考えられる．

4.6.4 $J_{\max}$を用いる学習

第3章で示したロバスト制御系の学習法において共通に用いられていた方針は，制御系にとって最悪な環境における制御性能を改善するというものであるが，確率的な不確かさに対してもその考えを適用することを試みる．最悪な環境における制御性能は統計量の一つである最大値を用いて表すことができるので，次の評価関数 (4.27) を最小化する学習方法が考えられる．

$$J_{max} = \max[J] \tag{4.27}$$

ここで，評価関数 (4.27) は微分不可能であるため勾配に基づく学習アルゴリズムは適用できないが，Powellの共役方向法を用いる学習アルゴリズムを用いることにより学習を行なうことが可能となる．また，学習を行う際には各統計量として標本抽出による推定量を用いるが，最大値の推定は標本数が小さいと十分な精度を持たないので，大きな標本数の集団で計算を行なう必要がある．しかし，標本数が多くなるにともない，計算に時間がかかるようになるため，計算時間短縮を目的とした次のアルゴリズムを用いて，(4.27) の最小化を行なう．

[J_{max}最小化学習アルゴリズム]

1. 適切に初期化されたニューラルネットにおいて，標本数Nからなる大標本集団S_Nにおいて，Jを最大にする標本の番号をi_0とする．標本集合S_0を

$$S_0 = \{i_0\} \tag{4.28}$$

とする．また，$k = 0$とする．

2. 評価関数 (4.29) を最小にするようにネットワークの学習を行なう．

$$\max_{S_k} J \tag{4.29}$$

学習後のネットワークをnet_kとする．

3. S_Nにおいて，net_kの制御のもとでJを最大にする標本の番号をi_{k+1}とする．

4. $i_{k+1} \in S_k$であるならば，学習終了．そうでないならば，

$$S_{k+1} = S_k \oplus i_k \tag{4.30}$$

とする．ただし，$\oplus$は要素の追加をあらわす演算子であるとする．さらに，$k = k+1$として2へ戻る．その際，学習の初期状態はnet_kとする．

上記のアルゴリズムにおける (4.30) を (4.31) のように変更すると，勾配法に基づく学習アルゴリズムを適用することを可能となる．

$$S_{k+1} = i_k \tag{4.31}$$

しかし，この方法では最悪な標本が周期的に入れ替わる計算ループに陥いることを防ぐことができないために計算効率が悪くなる．また，最悪な場合には無限ループに陥り学習が完了しないことが予想される．

4.7 ロバスト制御系の学習の数値例

本節では，4.6節で述べた方法により風に対してロバストな高度制御系を学習した例を示す．他の制御入力に関する制御は適当に定めた線形制御系とし，全てのシミュレーションにおいて同一のものを用いた．他の制御が十分に上手く行なわれているならば，高度の学習に大きな影響を及ぼすことはなかった．飛行における評価関数 Jは次のように定めた．

$$J = \sum_{t=0}^{T} (z(t) - d(t))^2 + v_z^2(t) \tag{4.32}$$

ここで $z(t)$，$v_z(t)$ は高度および高度方向の速度であり，目標高度 $d(t)$ は次の通りであるとした．

$$d(t) = \begin{cases} 3.0 & (0 \leq t < 1) \\ 5.0 & (1 \leq t < 20) \\ 3.0 & (t \geq 20) \end{cases} \tag{4.33}$$

ネットワークへの入力は $z(t)$,$v_z(t)$ の 2 つとし，出力はコレクティブピッチ操舵角δ_cとする．また，中間層ユニット数は 7 個とした．

計算に用いたフライトシミュレータではランダム風はその大きさを 3 段階で選ぶことができるほか，乱数の種の変更が可能であった．そこで，ランダム風の乱数の種を変更することにより複数の飛行の標本を作成した．また，水平風と上下風の選択が個別に可能であったが，高度制御に影響の大きい上下風のみ最大となるように設定し，水平風は加えないとして学習を行なった．図 4.18に，フライトシミュレータで生成されたランダムな上下風の風速の一例を示す．学習の評価関数にあらわれる Jの平均や分散などは，このように作成した標本集合からの推定値を用いた．

ニューラルネットワークの学習による制御系の構築を行なう前に，線形制御系を用いて検討を行う．線形制御系として次のような微分先行型 PD 制御系を選び，比例ゲイン K_p，微分ゲイン K_dを様々に変更したときの Jの平均と分散を求めた結果を図 4.19に示す．

$$\delta_c(t) = K_p \cdot [d(t) - z(t)] - K_d \cdot v_z \tag{4.34}$$

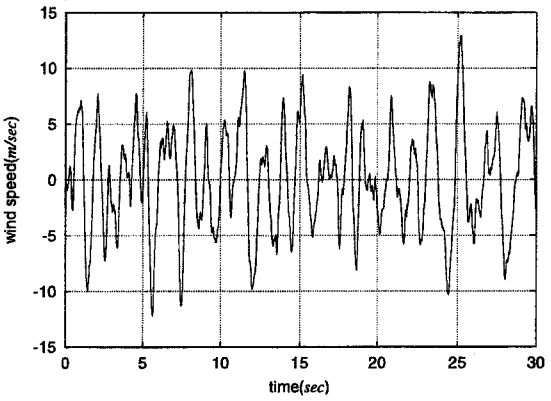


図 4.18: Velocity of a vertical random wind

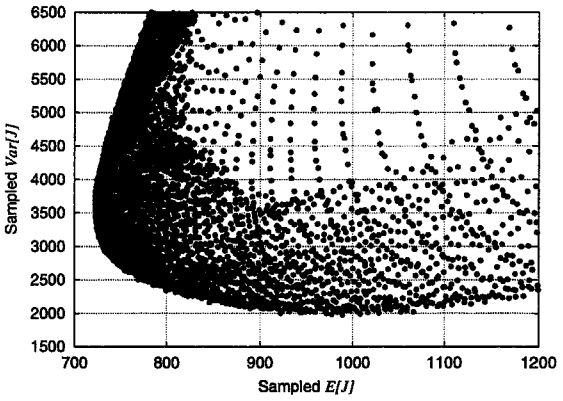


図 4.19: Sampled averages and variances of various linear controllers

各ゲインは 0.1 から 9.9 まで 0.1 の刻み幅で変更し計算を行なった．平均と分散には，標本数が 40 である標本集団から計算した標本平均と不偏分散を用いている．図 4.19は良好な制御結果であった領域を拡大しており，ここで調べた範囲内にあるゲインの組み合わせでは，高度を安定に保つことができないものも多く存在した．図 4.19より明らかなように，同じ平均であるが異なる分散を持つ制御系が多数存在することが分かる．また，4.6.2節にて述べたように，平均最小かつ分散最小という多目的最適化問題には多数のパレート最適解が存在することも分かる．もし，(4.34) のような形の制御系を用いるとすると，図 4.19に示されているパレート最適解の一つを制御系として選べばよいことになる．

4.7.1 J_γを用いる学習

本節で示す学習例では，標本数を 20 とする標本集合から計算した標本平均，不偏分散を用いる．ここでは，評価関数として (4.23) を用いる．γを 0.0，0.05，0.5 と設定し，適当な乱数を用いて初期化された複数のニューラルネットワークを用いて学習を行った．学習が完了したネットワークに 1000 通りの乱数の種から発生させた上下風を適用し，計算した Jの平均や分散を図 4.20に示す．ここでは比較のために，風を全く考慮せずに (4.32) を用いて学習したニューラルネットワーク，すなわち最適制御系の学習を行ったニューラルネットワークによる制御結果も同時に示す．図 4.20より明らかなように，風を考慮せずに学習を行なったニューラルネットワークでは，平均，分散ともに大きなものとなっている上に，学習結果のばらつきが大きく，好ましい結果が得られていないことが分かる．この結果も，第2章のようにロバスト性を考慮せずに最適フィードバック制御系を学習したニューラルネットワークのロバスト性は全く予測できず，ロバスト性の観点からいって好ましくないことを示していると思われる．これ対して，4.6.2節で提案した手法を用いて学習を行なったニューラルネットワークは，平均・分散ともに小さくなっている上に，学習結

果のばらつきが非常に少なくなっており，良好な結果が得られていることが分かる．また，γの増加につれて，Jの分散が小さくなっていくとともに，Jの平均は大きくなっていっていることも分かる．図 4.21に各制御系におけるJの分布をヒストグラムを用いて表わす．この図を見ると明らかに，γの増大にしたがってJの分布する範囲が狭くなり，Jの分布は平均値付近に集中するようになるが，それに伴って平均は徐々に増加していく様子が分かる．また，風を考慮せずに学習を行なった制御系によるフライトの例と$\gamma = 0.5$として学習を行なった制御系によるフライトの例を図 4.22および図 4.23に示す．それぞれの図中には各制御系のおける無風状態における制御結果も示した．図 4.22に示した無風状態における制御結果によると最適制御系を学習したニューラルネットワークによる制御では，下降と上昇の場合では運動特性はあまり大きな違いは見られないことが分かる．これに対して，図 4.23における無風状態の制御結果により，γを増加させることによりロバスト性の要求を大きくすると，下降と上昇では違った運動特性を持つようになるということが分かる．それぞれのニューラルネットワークによる風のある場合のシミュレーションの結果より，フライトシミュレータにおいて風がRMAXに与える影響は上下方向に対称ではなく，明らかに高度を下げるという影響が大きいことが分かる．[54,55] によると，メインロータに吹き込む上下風の風速によりその流れの様子は分岐するので，推力は風速に対して非線形な変化をすることが記載されているが，フライトシミュレータにはそのような効果が入っているものと思われる．このために，ロバスト性を全く考慮しない場合には上昇・下降にとってほとんど対称であった制御が，ロバスト性を考慮したために非対称な制御へと移行していったと考えられる．上昇・下降において非対称な制御が上下風に有効になることを示すために，(4.34) で表されたPD制御系と，(4.35) のように各時刻における目標高度との差の符号による切り換えにより上昇と下降では異なるゲインを持つPD制御系による制御結果を比較する．

$$\delta_c(t) = K_p\left(sgn\left(d(t) - z(t)\right)\right) \cdot \{d(t) - z(t)\} - K_d\left(sgn\left(d(t) - z(t)\right)\right) \cdot v_z \tag{4.35}$$

ここでは，1000通り上下風の標本集合に基づいて，計算したJの平均と分散を図 4.24に示す．ただし，ここではパレート最適解のみを示している．さらに，パレート最適解となる各制御系の各ゲインとJの平均や分散の関係を図 4.25に示す．図 4.24より，(4.35) のような簡単な非線形な制御系であっても，ロバスト性，制御性能ともに線形制御系よりも優れることが明らかである．また，(4.35) の制御系では，下降に比べると上昇の比例ゲインが大きくなっており，ニューラルネットワークが学習により得た制御系と同じ特性となっていることが分かる．また，図 4.24には線形制御系 (4.34) と非対称なゲインを持つ制御系 (4.35) のそれぞれのパレート最適解とともに，提案した手法により学習を行なったニューラルネットワークによる結果が同時に示されているが，ニューラルネットワークによる制御はいずれのパレート最適解を大きく上回る結果となった．図 4.24のパレート最適解を求めるために，標本数40である小規模の標本集合における推定平均と不偏分散を計算した結果を利用するなど，パレート最適となるゲインの存在範囲を絞って計算を行う

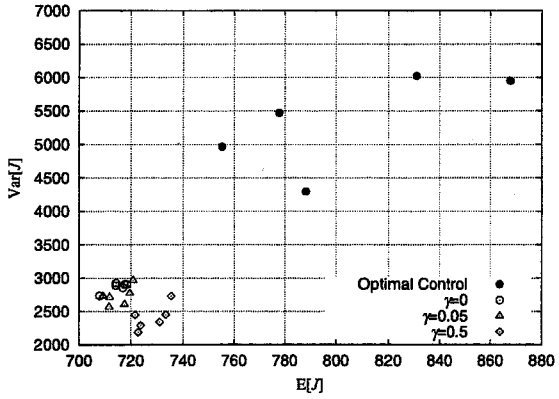
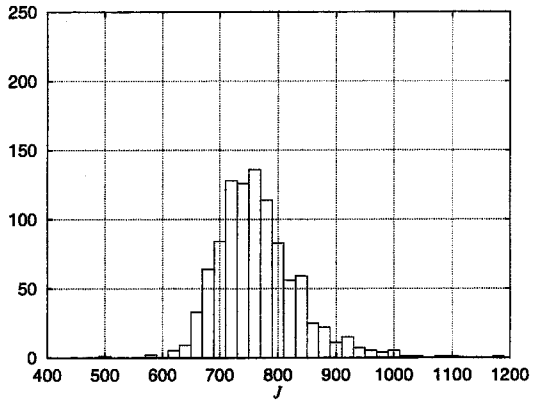
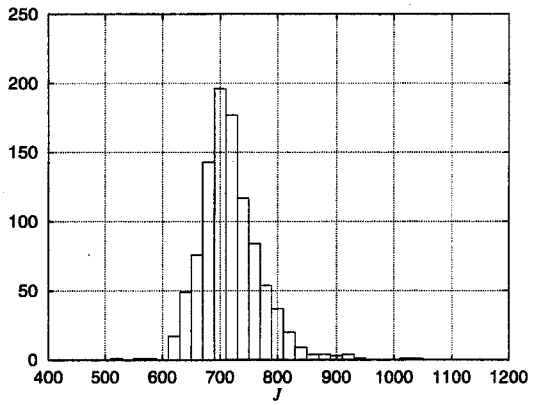


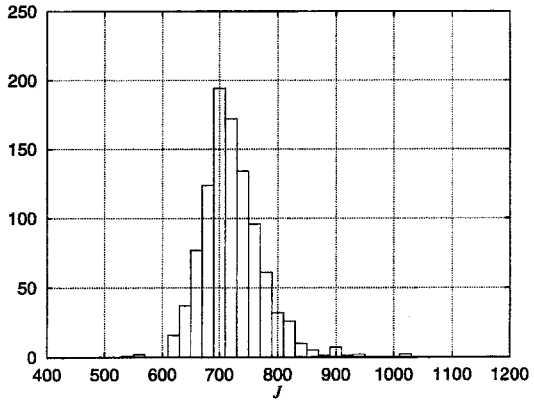
図 4.20: Average and variance of J



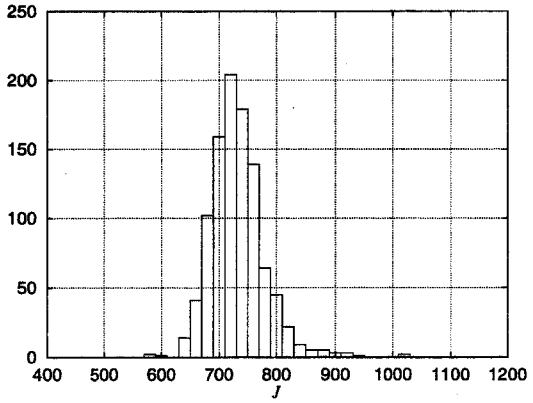
(a) Optimal control



(b) $\gamma = 0$



(c) $\gamma = 0.05$



(d) $\gamma = 0.5$

図 4.21: Distribution of J

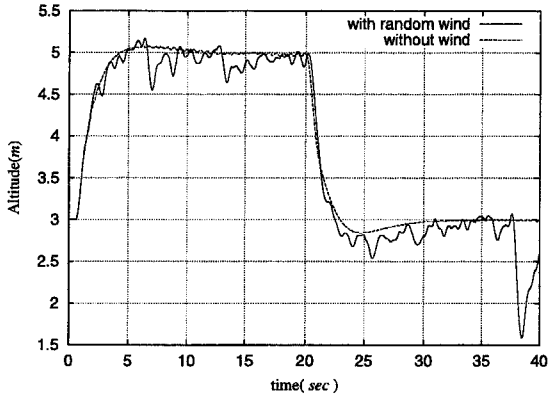


図 4.22: Altitude control(Optimal controller)

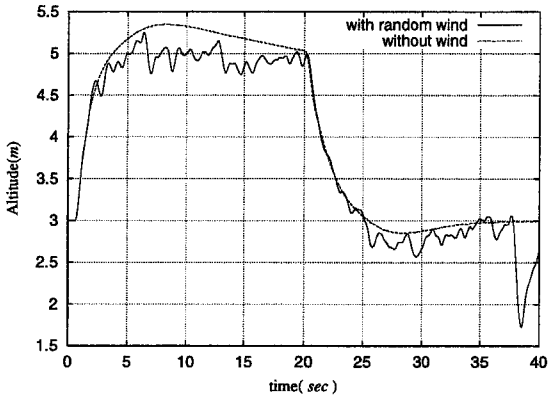


図 4.23: Altitude control($\gamma = 0.5$)

など，計算時間の短縮を図っている．しかし，AMD Athlon 900MHz の CPU を持つ PC を用いたとき，線形制御系 (4.34) のパレート最適解を求める計算にはおよそ 20 時間が必要であった．非対称なゲインを持つ制御系 (4.35) ではゲインの組合せ数が線形制御系 (4.34) におけるそれに比べて非常に多くなるために，複数の PC を用いて分散的に計算を行なったが，AMD Athlon 900MHz の CPU を持つ 1 台の PC を用いるとすると，1ヶ月以上の計算時間が必要であった．(4.34) あるいは (4.35) で表される制御系のゲインをフライトシミュレータのみから発見的な手法を用いて求めるには，このように大規模な計算時間が必要となり，非常に効率が悪い設計法といえる．これに対して，ニューラルネットワークの学習に要した時間は 15 から 20 時間であった．このように計算に必要な時間，得られる制御性能，ロバスト性などすべての面でニューラルネットワークの学習を用いる方法が勝っており，優れた制御系の設計法といえるだろう．

4.7.2 J_vを用いる学習

本節で示した学習例でも，前節と同様に標本数を 20 とする標本集合から計算した標本平均，不偏分散を用いる．Jの平均値を用いた拘束条件にあらわれる E_vを 690，700，710，720，730，740，750，800，1000 とした．ただし，E_vが 720 以下では等式拘束条件 (4.25) を，730 以上では不等式拘束条件 (4.26) を用いた．拘束条件を学習に組み込むために，Lagrange 乗数λ，μを用い，拡張 Lagrange 関数 L(4.36) を最適化することにより学習を行った [22,56].

$$L = J_v + \lambda(E[J] - E_v) + \mu(E[J] - E_v)^2 \tag{4.36}$$

図 4.26に学習の際に用いた標本による Jの推定平均と，1000 個の標本数からなる標本集合を用いて計算した Jの平均を示す．この図によると，学習において E_vにより制御性能を設定することができていることが分かるが，大きな標本集合で計算した平均に比べると学習に用いた推定平均は必ず小さくなっていること

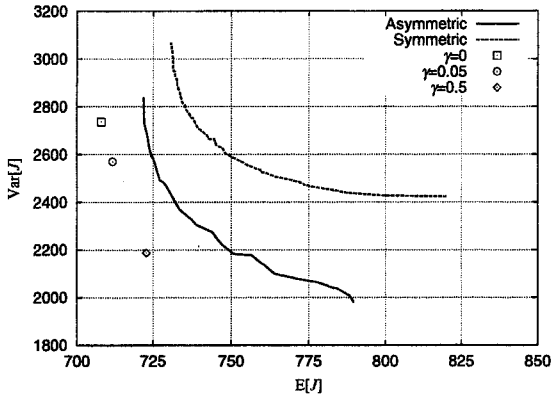
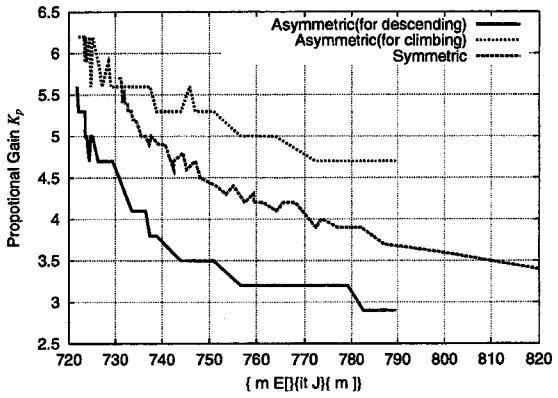
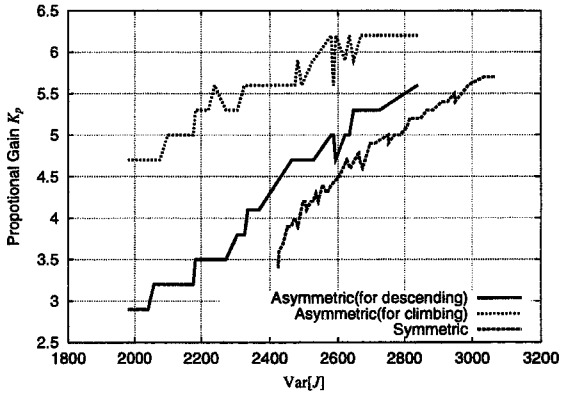


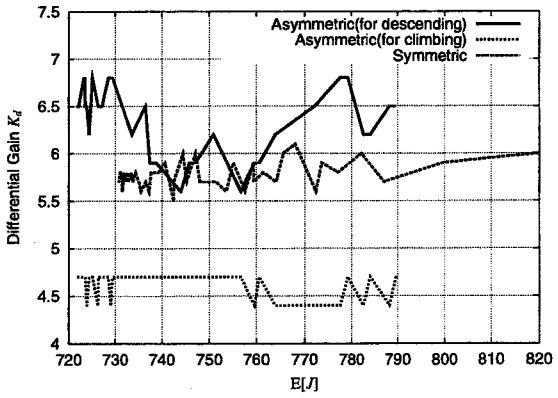
図 4.24: Pareto optimal solutions and results of trained neural networks



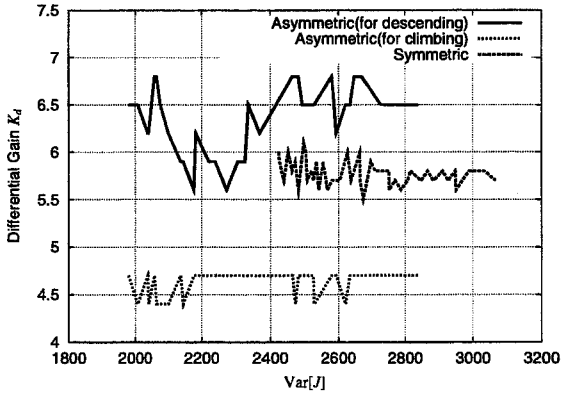
(a) Propotinal gain and averages



(b) Propotinal gain and variances



(c) Differential gain and averages



(d) Differential gain and variances

図 4.25: Pareto optimal gains of PD controller and asymmetric PD controller

が分かる．これは図4.21に示したJの分布から分かるように，Jの分布は右に片寄っていることや，学習が用いた標本集合に特殊化してしまうことなどが原因であると思われる．図4.21の分布の歪度や尖度などを調べると正規分布に比べて分布の広がりが小さく，かつ右にすそが伸びた分布になっていることがわかる．また，複数の初期ネットワークから学習を行ない，Jの平均と学習に用いた標本を用いて計算したJの平均の相関を求めた結果を図4.27に示す．図4.26および図4.27より，学習に用いた平均の推定値には一定の偏差が存在するが，平均の真値との相関は高いことが分かる．このため，Jの平均の推定に偏差を生じているが，学習における推定平均と実際の平均の相関は高いので，E_vにより正しく学習後の制御性能を設定することが可能であることが分かる．また，学習の際に$E_v < 700$のように選択しても，E_vの等式拘束条件を満たす制御系を学習により得ることができなかったが，これは制御入力など制御系の限界のためであると考えられる．$E_v > 700$ときには等式拘束条件をほぼ満たす制御系を学習により得ることができた．また，$E_v < 800$までは不等式拘束条件を課したときであっても，拘束条件は$E[J] = E_v$となって満足されていた．これに対して十分大きなE_vを用いると，拘束条件は$E[J] < E_v$となって満足されていた．これは，図4.19に示したように，Jの平均がある値以上より大きくなると，Jの分散の最小値はかえって大きくなるためである．よって，分散を最小にする安定な制御系を学習するためには，不等式拘束条件を用いる必要がある．等式拘束条件を用いるときに，E_vをある一定値より大きくすると，制御性能・ロバスト性能ともに好ましくない制御系が学習の結果として得られるので注意が必要である．図4.29に学習が完了したニューラルネットワークによる制御結果を示す．この図は，上下風の乱数の種を1000通り変更したときのJの平均や分散の関係を示しているが，E_vの増加にともない$Var[J]$は小さくなる傾向が見られる．しかし，一部の結果は$E[J]$と$Var[J]$，すなわちロバスト性との関係の考察に反して，$E[J]$が増加するとき$Var[J]$も増加している．この原因としては，学習が局所解に陥ったことがまず考えられる．図4.28に複数の初期ネットワークから学習を行なった結果を示す．この結果を見ると，Jの推定平均に関しては初期ネットワークにより大きな差が生じていないのに対して，不偏分散には大きな差を生じていおり，多数の局所解が存在していることが分かる．このように局所解に陥る現象は，2章や3章などで示した数値例ではほとんど見られなかった．よって，学習アルゴリズムに原因があるとは考えられない．この原因としては，計算に用いたフライトシミュレータにあると考えられる．このフライトシミュレータでは，飛行状態を表す各変数は実際のRMAXの制御用と同じ書式を用いるために，浮動小数点型ではなく整数型を使っている．例えば，高度など位置情報はcm単位以下は四捨五入されている．このように変数の書式が整数型であるために学習が局所解に陥ったと考えられる．また，このほかの原因として考えられるのは，学習に用いた標本数の不足による分散の推定誤差が挙げられる．図4.27には学習に用いた標本から推定した分散と大きな標本集合から計算した分散との相関係数を計算したものも示されているが，大きなE_vを学習に用いてロバスト性が高い制御系の学習

を行ったとき，不偏分散と真の分散の相関が低かったことが分かる．これは，ニューラルネットワークが学習に用いた標本集合に特殊化して学習を進めてしまったからである．このために，ロバスト性を重視した設計を行なうには分散の推定を改善し，標本集合に特殊化した学習を行なわないようにする必要がある．

4.7.3 J_{max}を用いる学習

本節では，学習に用いる標本数は1000とした．他の二つの方法に比べて標本の数が多いが，4.6.4節で示した方法により，学習が速やかに終了することを期待することができる．実際に，他の二つの方法に比べてかなり短い時間で学習が終了する．ここで，4.6.4節で示した方法において(4.30)を(4.31)のように変更すると，いくつかのランダム風の乱数の種の間をループしてしまい，学習が完了することはほとんどの場合において不可能であった．

複数の初期状態から学習を行なったネットワークによる制御のもとで，上下風の乱数の種を1000通り変更したときのJの平均，最大値および分散を図4.30，4.31に示す．図4.30より，Jの平均および最大値は初期化に依らずほぼ同じ値に収束していることが分かる．また，図より$J_{\max}$を最小にするようなネットワークによるとJの最大値は9.830×10^2となった．実際に，Jの他の一次のモーメント量におけるネットワークの初期化の影響は小さなものであった．しかし，一次のモーメント量に比べると，二次のモーメント量である分散のばらつきが大きいことが図4.31により明らかである．高次のモーメントである歪度なども，学習の初期条件によるばらつきが大きいことが確認されている．また，図4.31とJ_γやJ_vを用いた学習の結果と比較すると，本手法により学習を行なったネットワークによるJの分散が大きくなっていることが分かる．これは本学習法においてJの高次のモーメントが評価の対象にならないことが原因であると考えられる．このようにJの分散を小さくすることはできていないが，この方法ではJの最悪劣化を考慮するため，制御系の評価はもっとも安全であると考えられる．このために，他の方法と比べると，本学習法により得られる制御系は最も信頼性が高いといえる．

4.8 各学習法の比較と改善法

本節では，4.6節において提案した方法の特徴を検討を行ない，特性を改良する学習について考察する．

J_γを用いる学習では，ロバスト性を定量的に評価することが可能である．確率的な不確かさに対するロバスト性の大きさは，学習に用いる評価関数(4.21)や(4.23)におけるγの大きさにより指定することができる．特に，$1/\gamma$は確率的な不確さから評価出力へのH_∞ゲインとなり，3.3でも用いたように一般的に用いられているロバスト性の定量的評価であるために，非常に有用な方法となる．ロバスト性を最大限に高める学習を行うには，γの最大値を知る必要があるが，このためには学習を繰り返して行なうことにより，γの

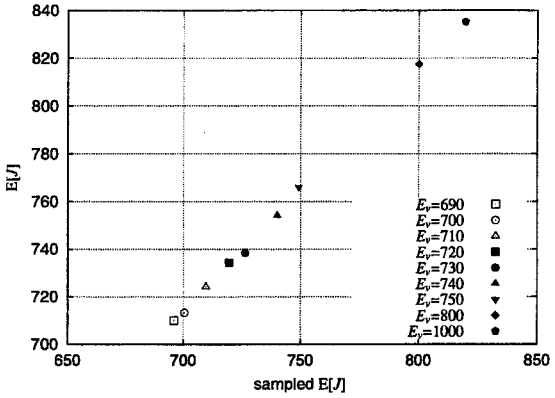


図 4.26: Sampled $E[J]$ used in training and $E[J]$

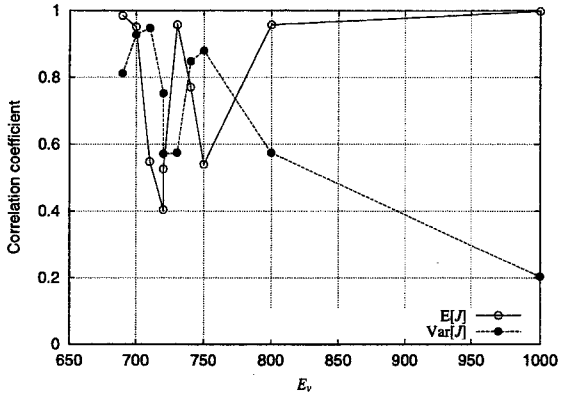


図 4.27: Correlation coefficients

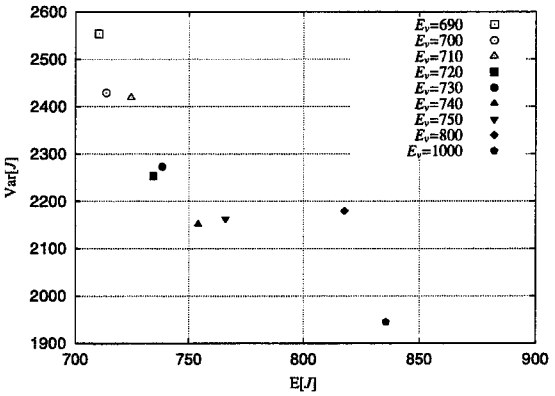


図 4.28: $E[J]$ and $Var[J]$

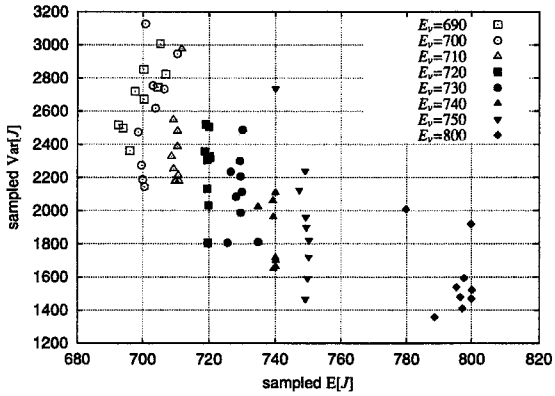


図 4.29: sampled $E[J]$ and sampled $Var[J]$

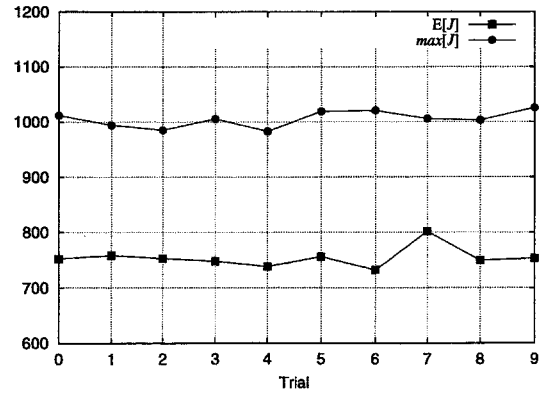


図 4.30: $E[J]$ and max$[J]$

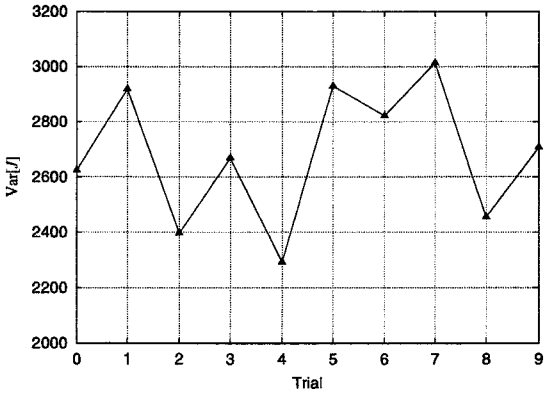


図 4.31: $Var[J]$

最大値を探索する必要がある．しかし，厳密にロバスト性の最大値を求める要求は実際にはほとんどないと思われるので，この欠点が問題となることは少ないであろう．γの大きさにより，ロバスト性と制御性能のトレードオフを行なうことが可能であるが，制御性能を学習の際に指定することができないために，得られた制御系の制御性能が十分でないことが考えられる．よって，J_γを用いた学習はロバスト性を重視して制御系の設計を設計することが容易であるが，制御性能を重視した設計を行なうことは難しいといえる．

J_{max}を用いる学習には，Jの分散を小さくすることは評価関数に含まれていない．このためにJの最大値は小さいにも関わらず，Jの分散が大きくなる場合があることは，図4.31より明らかである．しかし，確率的な不確かさによる最悪の制御性能を考慮して学習を行うために，学習が完了したニューラルネットワークによる制御は最も安全なものとなる．このために，ロバスト性を最も重要と考える際に用いるべき方法であろう．また，4.6.4節に示したアルゴリズムのために，大きな標本数を用いても，他の二つの学習法に比べると計算時間がかなり短くなっているので，いくつかの初期点から学習させた中で，最もJの分散が小さくなったものを選ぶことにより，さらに良好な結果が得られるであろう．しかし，この方法のようにロバスト性を最大限にするとき，保守的な制御結果が得られやすいので注意が必要である．また，この方法ではロバスト性と制御性能のトレードオフを行なうこともできない．しかし，4.6節で示した他の学習法と複合して用いることにより，さらに優れた学習法則を導くことできる可能性がある．例えば，Jの最大値の上限J_mを設定し，

$$\max[J] - E_m \leq 0 \tag{4.37}$$

のような不等式条件をJ_γを用いる学習やJ_vを用いる学習において付加する方法が考えられる．ここでは，J_vを用いる学習において不等式拘束条件(4.26)を(4.37)のように変更して学習を行なった例を示す．4.6.4節で述べたように，最大値の推定を大きな標本集団で行なうために，次のような学習アルゴリズムを用いる．

[J_v学習アルゴリズム(最大値拘束条件付き)]

1. 適切に初期化されたニューラルネットにおいて，標本数Nからなる大標本集団S_Nにおいて，Jを最大にする標本の番号をi_0とする．標本集合S_0を

$$S_0 = \{i_0\} \tag{4.38}$$

とする．また，$k=0$とする．評価関数Jの分散を推定するために用いる標本集合をS_vとする．ここで，S_vは適当な個数の標本からなるとする．

2. 不等式拘束条件(4.39)付きでS_vから算出される不偏分散を最小化するように学習する．学習後のネッ

トワークを net_k とする．

$$\max_{S_k} J - E_m \leq 0 \tag{4.39}$$

3. S_N において，net_k の制御のもとで J を最大にする標本の番号を i_{k+1} とする．

4. $i_{k+1} \in S_k$ であるならば，学習終了．そうでないならば，

$$S_{k+1} = S_k \oplus i_k \tag{4.40}$$

とする．ただし，$\oplus$ は要素の追加をあらわす演算子であるとする．さらに，$k = k+1$ として 2へ戻る．その際，学習の初期状態は net_k とする．

ここで，2. における学習は，不等式拘束条件 (4.39) 付き最適化問題となるが，Lagrange 乗数λ，μを用いて，拡張 Lagrang 関数 L(4.41) を最適化することによって行なう．

$$L = J_v + \lambda \left(\max_{S_k} J - E_m \right) + \mu \left(\max_{S_k} J - E_m \right)^2 \tag{4.41}$$

ここで示す数値例では，S_vに含まれる標本数を 20 とし，最大値を推定するために用いる標本数 Nを 1000 とした．$E_m = 990$ と設定し，学習を行なったネットワークによる制御結果を図 4.32 および図 4.33に示す．4.7.3節において $\max[J]$ の最小値は 9.830×10^2であったが，この値より小さな E_mを満足させることはできないために，ここでは $E_m = 990$ と設定した．図 4.32を見ると，設定された拘束条件を満足するように学習していることが分かる．また，4.7.3節で示した結果とを比べると，本節において示した方法は分散が小さくなっていることも分かる．4.7.3節において得られた最適なネットワークによる Jの分散は 2.292×10^3であった．この結果と，図 4.33より，Jの最大値はほぼ同じであるにもかかわらず，Jの分散が小さくなるニューラルネットワークが得られたことが分かる．さらに，E_mを 1000，1100，1200，1300 と設定したときの学習に用いた標本集合による Jの推定平均・不偏分散と，大きな標本集団を用いて計算した推定平均・不偏分散との相関係数を図 4.34に示す．この図によると，E_mの値によらず Jの平均の相関は高いことが分かるが，E_mの増加にともない分散の相関がほぼ 0 となり，学習に用いた標本集合から計算した不偏分散と真の分散はほぼ無相関になってしまうことが分かる．よって，ロバスト性を重視した学習を行なう際には，分散の推定が重大な問題となることが示された．

既に述べた二つの学習法に比べて，J_vを用いる学習では制御性能を重視することが容易である．他の学習方法ではロバスト性を定量的に評価することや，最大限にすることに重点がおかれており，制御性能をあまり重視しないために学習の際に Jの平均，すなわち制御性能を設定することができない．これに対して，本学習法では制御性能を学習の際に設定することが可能であり，要求される制御性能を持つ制御系の設計を行うことが可能である．しかし，4.7節で示した数値例によると，学習のときに用いた標本に基づく推定平

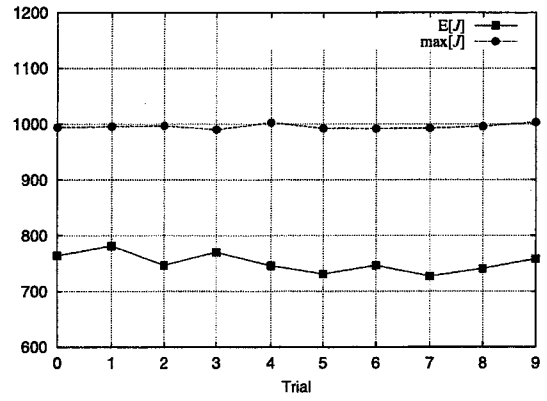


図 4.32: $E[J]$ and $\max[J]$

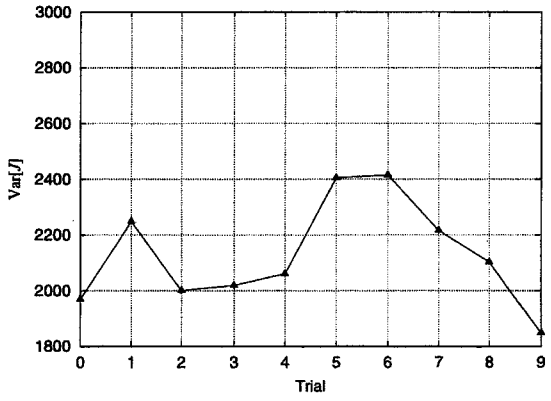


図 4.33: $Var[J]$

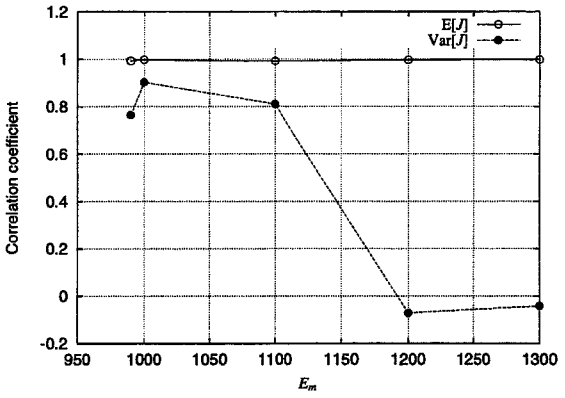


図 4.34: E_m and correlation coefficients

均と比べると，Jの平均は必ず大きくなっていることが分かった．このため，学習において設定した制御性能を厳密に満足しているとはいえなかった．また，学習に用いた標本集合から計算された分散と大きな母集団により計算した分散の相関を調べると，E_vを大きくするにつれて小さな値となっていくことが分かった．すなわち，学習後のネットワークのロバスト性を向上させようとすると，分散の相関が小さくなってしまった．これは，ニューラルネットワークが少ない標本数からなる標本集合において各標本に存在する特性に特殊化した学習をおこなってしまったことを意味する．よって，本方法では平均および分散のさらに高精度な推定方法が必要であると考えられる．このように限られた標本集合に特殊化してしまうことは非常に危険であるので，特に信頼度の高い分散の推定法が必要である．

まず，制御性能の推定法として次の二つの方法を考える．

1. 平均値の代わりに，ロバストな統計量といわれているメジアンを用いる．

2. 平均値の推定には区間推定を用いる．

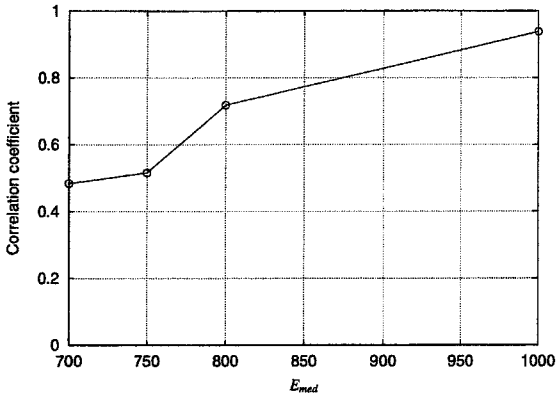


図 4.35: E_{med} and correlation coefficient

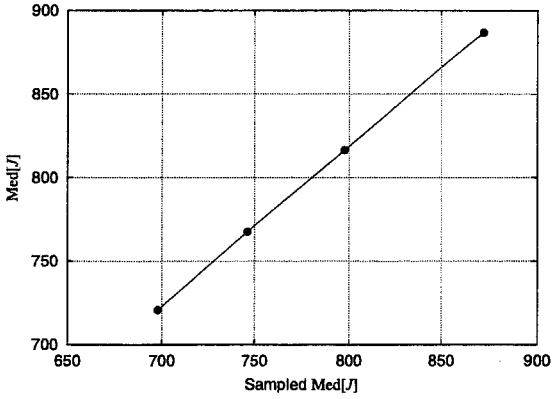


図 4.36: sampled $Med[J]$ and $Med[J]$

平均値の代りにメジアンを用いる方法は，Jのメジアンを $Med[J]$ と書くこととすると，拘束条件 (4.25), (4.26) を

$$Med\,[J] - E_{med} = 0 \tag{4.42}$$

$$Med\,[J] - E_{med} \leq 0 \tag{4.43}$$

のように置き換えるだけでよい．明らかなように $Med[J]$ は微分不可能であるが，Powell の共役方向法に基づく学習アルゴリズムを用いるので，メジアンを用いても学習の実行は可能である．J_vによる学習において，拘束条件 (4.43) を用いた例を示す．ここでは，E_{med}を 700，750，800，1000 の 4 通りに変更し，学習を行なった．$E_{med} \leq 800$ としたとき，付加した不等式拘束条件は等式として満足されていたが，$E_{med} \leq 1000$ としたとき，学習に用いた標本から計算したメジアンはおよそ 875 であり，拘束条件は不等式となって満足されていた．図 4.35に，学習に用いた標本集合から計算したメジアンと大きな母集団から計算したメジアンとの相関係数の関係を示す．E_{med}が小さいときには相関が少し小さくなっているが，学習に用いた標本から計算したメジアンと実際のメジアンは十分に高い相関があることが分かる．さらに，図 4.36に学習に用いた標本集合から計算したメジアンと大きな母集団から計算したメジアンを示す．この図ではいくつかの初期ネットワークを用いて学習を行なった結果，学習後のネットワークの分散が最も小さくなったネットワークにおけるメジアンを示している．明らかなように，学習のときに用いられた標本に基づくメジアンは真値に対して常に小さくなってしまっている．これは 4.7節において示した結果と同じものである．メジアンは異常値に対してロバストな推定量といわれているが，学習に用いた標本集団では平均値と同じようにメジアンも必ず小さく評価されるということは，異常値により推定の誤差が生じているというわけではないことを示している．よって，この場合はメジアンを用いても，平均を用いた場合と同じような結果となった

統計量の点推定に対して，区間推定とは統計値の真値がある区間 $[E_L, E_U]$ に入る確率を $(1-\alpha)$ 以上と

なるように保証する方法である．標本数を n とすると Jの平均の信頼係数 $(1-\alpha)$ の信頼区間は

$$\left[\bar{J} - \frac{t_\alpha(n-1)\cdot s}{\sqrt{n}}\ ,\ \bar{J} + \frac{t_\alpha(n-1)\cdot s}{\sqrt{n}}\right] \tag{4.44}$$

ここで，$t_\alpha(n)$ は t 分布表のパーセント点から求められる．また，J_iを標本 i において計算した評価関数の値とすると，$\bar{J}$と s は点推定による平均および標準偏差の不変推定量である．

$$\bar{J} = \frac{1}{n}\sum_{i=0}^{n} J_i \tag{4.45}$$

$$s^2 = \frac{1}{n-1}\sum_{i=0}^{n}\left(J_i - \bar{J}\right)^2 \tag{4.46}$$

以上から，$J_{trust}(1-\alpha)$ を次式のように定めると，

$$J_{trust}(1-\alpha) = \bar{J} + \frac{t_\alpha(n-1)\cdot s}{\sqrt{n}} \tag{4.47}$$

Jの平均が $J_{trust}(1-\alpha)$ 以下である信頼係数は $(1-\alpha/2)$ であり，$J_{trust}(1-\alpha)$ を

$$J_{trust}(1-\alpha) - E_c \leq 0 \tag{4.48}$$

のように学習の拘束条件として用いることで，$(1-\alpha/2)$ の信頼度で制御性能が E_c以下となる学習方法となる．ここでは，$\alpha = 0.02$，標本数を 20 として計算を行なった例を示す．拘束条件 (4.48) における E_cを 730，740，750，800 と変化させたとき，$J_{trust}(1-\alpha)$ と大きな標本数を用いて計算した $E[J]$ を比較した結果を図 4.37に示す．ただし，図中には複数の初期ネットワークから行った学習の結果を示した．$E_c = 800$ としたときの結果には多少のばらつきが見られるが，その他の E_cのときの結果のばらつきが少なく良好な結果といえる．また，ニューラルネットワークは (4.48) に示した拘束条件を満足するように学習を行なっていることが分かる．また，$J_{trust}(1-\alpha)$ と $E[J]$ を比較すると $J_{trust}(1-\alpha)$ は $E[J]$ を上から抑える評価となっているため，学習において制御性能が過大に評価されないため，安全な性能評価基準となっていることが分かる．このように (4.48) のように $J_{trust}(1-\alpha)$ を拘束条件として用いることにより，J_vを用いる学習において制御性能の設定の信頼性を向上させることが可能となる．学習に用いた標本集合から計算した推定平均，不偏分散と大きな標本集合から計算した平均，分散との相関係数を図 4.38に示す．図から明らかなように，推定平均は平均と高い相関を示したが，E_cが大きくなると不偏分散と分散との相関が小さくなった．この方法においてもロバスト性を重視した学習を行なう際には，分散の推定が重大な問題となることが示された．

次に，分散の推定の信頼度を上げる方法について考察する．分散の信頼度 $(1-\alpha)$ の区間推定は

$$\left[\frac{(n-1)s^2}{\chi^2_{\alpha/2}(n-1)}\ ,\ \frac{(n-1)s^2}{\chi^2_{1-\alpha/2}(n-1)}\right] \tag{4.49}$$

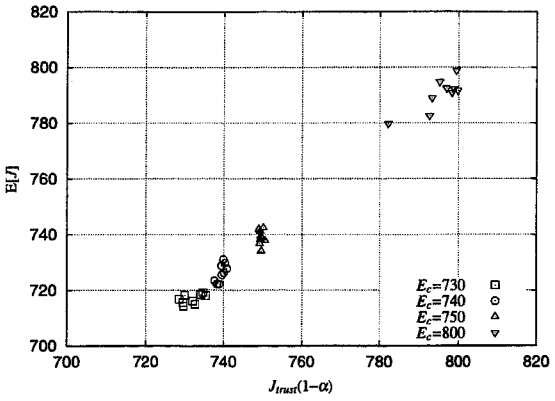


図 4.37: $J_{trust}(1-\alpha)$ and $E[J]$

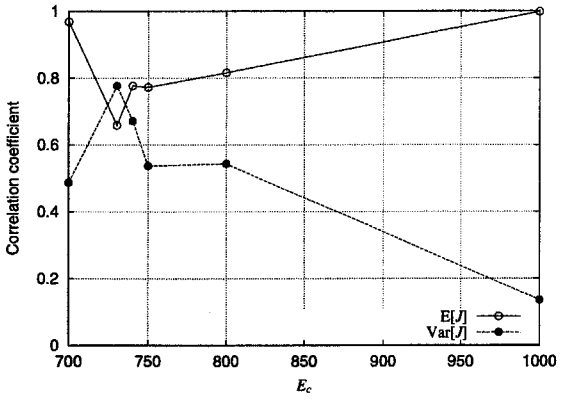


図 4.38: E_c and correlation coefficients

である．ここで，$\chi^2_{alpha}(n)$ はχ^2分布表から求めることができる．しかし，分散の区間推定の上限および下限はいずれも分散の不偏推定量の定数倍にすぎず，これらを学習に用いても，分散の不偏推定量を用いる方法と同じ結果となる．このため，分散の推定の信頼性を向上させるにためには標本数を増やす以外に方法がない．図 4.39に標本の数と Jの不偏分散推定値の関係を示す．図中には次の 4 つ制御器の結果を示した．

- PD 制御 [A]$(K_p = 5.0, K_d = 5.5)$
- PD 制御 [B]$(K_p = 5.0, K_d = 5.5)$
- J_vを用いる学習を行なったニューラルネットワーク [C],[D]

ニューラルネットワーク [C],[D] は，異なる初期ネットワークから $E_v = 1000$ とした不等式拘束条件を付加した J_vによる学習を行なったものである．図 4.39において，標本数が 20 までの結果は学習に用いた標本から計算されている．学習後のニューラルネットワークは標本数が少ないとき，分散の推定値が実際の値より小さくなっている．このような特徴は PD 制御系では見られない．よって，これは標本集合に特殊化した学習を行なってしまっていることを意味する．標本集合への特殊化は，ロバスト制御系の学習にとって大きな問題となる．単純な方法であるが標本数を増加することにより，標本が多様となって特殊化が軽減されると考えられる．また，分散の推定の精度も向上すると考えられる．現在のところ，標本集合への特殊化に対応する方法としては，学習に用いる標本数を増やす方法以外のものが見つかっていない．学習に用いる標本数を増加することにより分散の推定が向上することを確認するために，学習に用いる標本の数をこれまでの 2 倍である 40 として学習を行なった結果を表 4.2に示す．この結果より，学習に用いる標本数を増加することにより，学習の標本集合への特殊化を軽減することが可能であり，推定分散と分散の相関も向上することが分かった．しかし，相関を高めることができたが，その推定精度は十分なものとは言えなかった．図 4.45に推定平均と平均，図 4.46に推定分散と分散を示したが，いずれの推定にも一定の偏差が

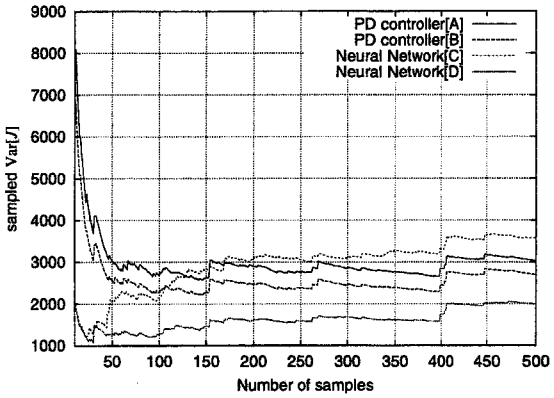


図 4.39: Relation between sampled $Var[J]$ and the number of samples

Table 4.2: Correlation coefficients of $E[J]$ and $Var[J]$ (samples=40)

Constraint	Correlation of $E[J]$	Correlation of Var[J]
$E_v < 800$	0.9952	0.5592
$E_v < 1000$	0.9964	0.9162
$E_c < 800$	0.9808	0.6087
$E_c < 1000$	0.9994	0.9392
$E_m < 1200$	0.9993	0.7294
$E_m < 1300$	0.9994	0.7679

あり学習に用いた推定値はいずれも真の値と比べると小さくなっていることが確認できる．これらの図によると平均の推定はそれほど大きな誤差ではなくなっていることが分かるが，分散の推定は十分なものとは言えないことが分かる．図 4.39から，分散の推定精度を十分に高めるためには，本研究で用いたフライトシミュレータでは 100 を越える標本数を用いる必要があるということが分かる．しかし，ここで用いている計算方法では標本数の増大は学習に必要な計算時間の増大となるので，あまり大きくすることができない．学習時間を現実的なものとするためには，さらに高速な CPU が必要である．必要とする CPU の速度を見積もるために，本研究で用いたフライトシミュレータを用いてある決まった学習を様々な計算機にて行い，学習に要した時間を測定した．計算機に使われている CPU が持つ計算能力を学習に要した時間の逆数として表すと，図 4.40示したような結果が得られた．図 4.40では AMD 社製の ATHLON1.4GHz の性能を 1 として表しているが，2001 年の夏に入手可能な CPU は，メーカによらず動作クロック数にほぼ比

例した計算能力をもつことが分かる．図4.40における最高性能を持つ計算機にて標本数を40として学習を行ったとき，学習方法や初期ネットワークにより差があるもののおよそ1日から2日の計算時間が必要であった．学習に用いる標本数をN倍にすると，学習に必要な時間もおよそN倍になるであろうから，おなじ時間で学習を完了するためにはCPUの動作クロックがN倍となる必要があることが判明した．しかし，動作クロック数の向上は物理的な制限により以前とは同様の向上速度とはならないと考えられている．よって，これまで用いたような計算方法では多大な計算時間を必要とすることになるため，計算時間の改善方法が必要である．各標本における評価関数の計算は独立・並行して行なうことが可能である．このために，複数台の計算機による並行計算が計算時間の短縮に有効であると考えられる．このような計算方法の有効性を示すために，フライトシミュレータにLANを媒介とする通信機能を付加することにより，図4.42に示したように複数のPCを結合させ，学習サーバであるPCが学習計算を制御し，学習クライアントPC群は学習サーバから与えられた情報に基づいて飛行シミュレーションを行い，得られた結果を学習サーバに送信するサーバクライアント方式のPC Clusterシステムを構築した．CPUの性能を計測したときと同様に，ある決まった学習をPC Clusterシステムを用いて行い，クライアントPCの台数とシステムの計算能力を求めた結果を図4.41に示す．この実験では，おなじ動作クロック数のPCをクライアントとして用いたが，図4.41によりクライアント数に比例して計算能力が向上することが分かる．構築したPC Clusterシステムでは，クライアントPCは同じCPUのものを用いねばならないということはない．CPUの性能に応じた計算配分を行うことも容易であり，すでに所有する処理能力の異なるPC複数台を用いて学習を行うことも可能である．PC Clusterシステムを用いた計算の有効性を示すために，J_vを用いる学習を行った結果を示す．ただし，ここでは$E_v \leq 1000$とし標本数を450とした．行った実験では10台から15台のPCを用いて学習を行ったが，学習に要した時間はいずれも24時間未満であった．これにより，PC Clusterシステムを用いることにより高速に学習が可能であることが示された．図4.43に学習に用いられた平均と実際の平均，図4.44に学習に用いられた分散と実際の分散を示す．また，図4.45および4.46に標本数が40とし学習したときの結果を比較のために示す．表4.2および図4.46より明らかなように，標本数が40とまだ十分でないときには学習に用いた推定分散と実際の分散には高い相関があるが，その推定には大きな差があったが，PC Clusterシステムを用いて標本数を450として学習を行ったときの結果では，学習に用いた推定分散はほぼ正しい推定値となったことが分かる．また，単一のPCを使った学習では得ることができなかった高いロバスト性を持つ制御系を構築することが可能であった．さらに，図4.43，および図4.45から，標本数が40であったときには実際の平均と推定値にはまだ一定の偏差が見られているが，PC Clusterシステムを用いて標本数を450として学習を行ったときの結果では，学習に用いた推定平均はほぼ正しい推定値となっていることが分かり，このために標本数が少ない場合よりもニューラルネットワークの学習が

進んだと考えられる．よって，確率的な不確かさに対するロバストな制御系の構築にはPC Clusterシステムによる学習が有効であることが分かる．

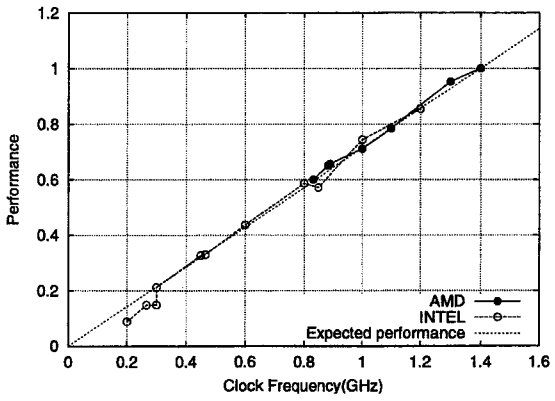


図 4.40: Performance of various CPUs

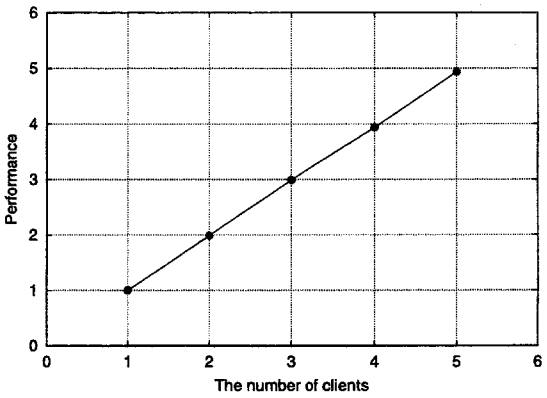


図 4.41: Performance of PC Cluster system

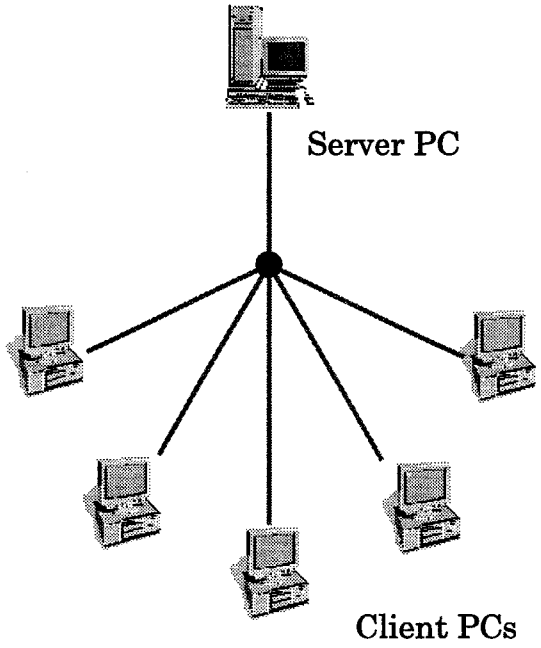


図 4.42: PC Cluster system

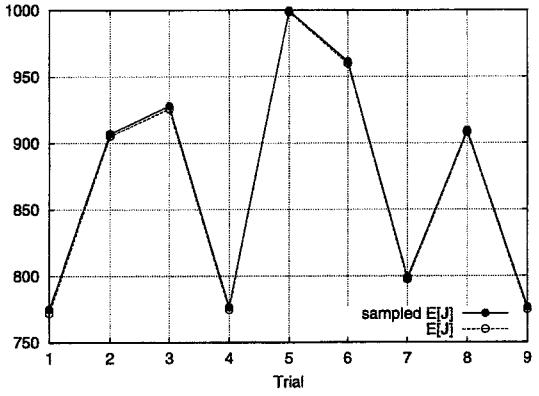


図 4.43: $E[J]$ (samples=450)

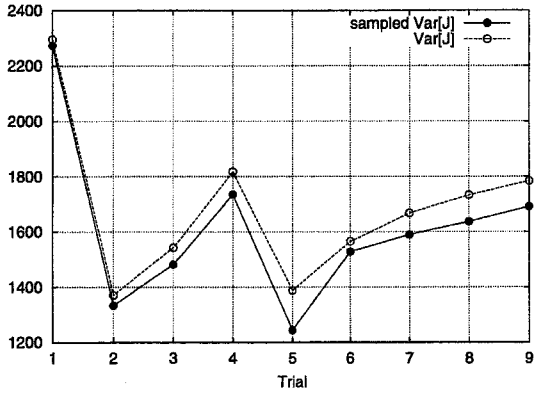


図 4.44: $Var[J]$ (samples=450)

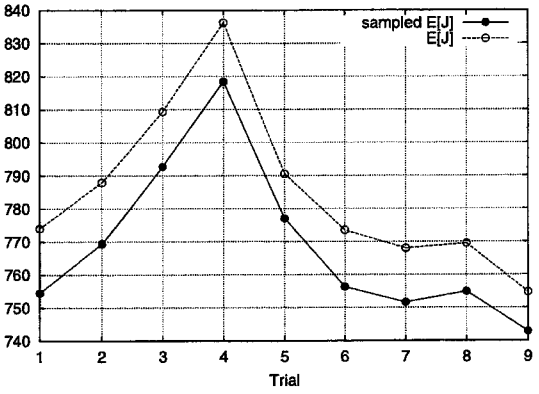


図 4.45: $E[J]$ (samples=40)

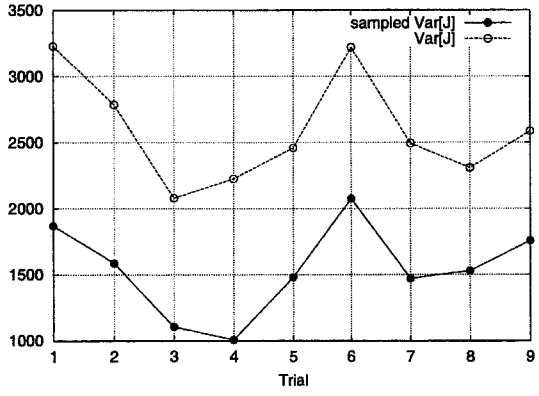


図 4.46: $Var[J]$ (samples=40)

4.9 結言

本章では，ニューラルネットワークの学習により，制御対象の情報を用いずに制御系を構築する方法を提案し，無人ヘリコプタ RMAX のフライトシミュレータにニューラルネットワークの提案する学習アルゴリズムを組み込むことで，自律飛行制御系の設計を行い，その有効性を示した．従来の方法では制御対象に関する詳しい知識が制御系の設計のためには必要であり，開発者には制御対象の詳しい知識が不可欠であった．しかし，UAV などの高度なシステムでは，全ての詳細な知識を制御系の設計者が持つことは困難である．高度なシステムにおいては，複数の専門家がそれぞれの部分に関する詳細な知識を持っているので，分散した知識を有効に制御系の設計に活用することが必要となる．また，UAV をはじめとして航空機の制御系の開発には飛行試験が不可欠であるが，コストの低減するためには飛行実験の回数を少なく抑えたい．このために，飛行データや専門家の知識の集大成としてフライトシミュレータの構築を行うが，本章において提案したニューラルネットワークの学習により制御系を構築する方法により，フライトシミュレータに組み込まれた様々な知識や実験結果を制御系設計の際に十分に活用することができる．また，全体のシステムを分散的に開発を行なうことが容易となるので，開発効率の向上につながると考えられる．このことは UAV の自律飛行制御系だけでは一般の対象における制御系の構築においても当てはまる．また，製品の特性パラメータやプラントの特性などは機密保持のために明らかにされていないことが多いが，対象の特性が公開することができない場合であっても，対象のシミュレータ，あるいはコンパイル済のオブジェクトファイルなどにより対象の情報が隠蔽されても，提案した学習方法を適用することが可能であるので，機密の漏洩のリスクは小さく押さえることができる．また，最近の米国において始まった研究に SEC(Software Enabled Control) があるが，高速な計算機とそれに伴い高度化するシミュレータ技術を利用して，適応あるいは学習機構を用い制御系を動的に構築しようとする試みであり，ハードウエアの一部をシミュレータと組合せて利用するなど，実機を用いた実験をできる限り少なくすることにより，開発に要するコストを下げるという目的も含んだ研究である．SEC は，ソフトウェアによって制御対象に適応することにより，制御系の設計や機能の向上を柔軟に行なうという考えに基づいているので，主としてオンライン学習が用いられている．しかし，オンライン学習を行なう際には，すでに学習済であるニューラルネットワークや何らかの方法で構築された制御系の存在が仮定されているが，[9,10] では天下り的に用意されたものを用いており，どのようにして学習済のネットワークを得るのかなどについては詳しく触れられていない．また，学習後のニューラルネットワークのロバスト性に関して，ほとんど考慮されていないため，十分とはいえないことも多い．本章で述べたように，オンライン学習はオフライン学習の結果を動的補正するために用いるべきであると考えられる．このため，提案したニューラルネットワークの学習による制御系の構築法を元に，ニューラルネットワークのオンライン学習を併用することで，制御系はよりフレキシブルでさらに高性能となる．

本章で示した飛行制御系の設計例では，簡単のために無人ヘリコプタの各軸まわりの運動は独立であると仮定していた．しかし，用いたフライトシミュレータにおいて運動の干渉が考慮されているために，シミュレーションにおいても各軸まわりの運動は干渉していた．しかし，各軸の制御系として適切なものを用いることにより，各軸を独立に制御をおこなっても，ほぼ干渉を打ち消すことが可能であったため，各軸の制御系を独立とした仮定は十分であるといえる．しかし，無人ヘリコプタの運動がさらに複雑となるとき，運動の干渉は無視できないものとなる．このため，今後の研究においてこのような仮定を取り除き．さらに高度な非線形制御の設計に発展させる予定である．例えば，旋回などは各軸を独立に制御する現在の方法でも可能であったが，その飛行は滑らかであるとはいえなかったが，滑らかな旋回を行なうためには，干渉を考慮し複数の制御入力を同時に制御するような非線形制御が必要となると考えている．

また，本章では確率的な不確さを伴うシステムに対してロバスト制御系を構築する方法を提案し，それらの手法を無人ヘリコプタの風の影響を軽減する高度制御系の構築に適用することによりその有効性を示した．ニューラルネットワークの学習により，上下方向の制御を非対称とすることが風による影響の軽減に有効であることが判明し，簡単なゲイン切換型 PD 制御系でもその有効性は確認することができた．しかし，ニューラルネットワークの学習により設計した制御系と同じレベルの制御系を発見的な手法では発見することは計算時間の点から考えると困難であり，計算時間からも提案手法が有効であることを示した．ヘリコプタの運動がそもそも非線形であり，従来より非線形制御系の適用が有効であると考えられている対象であったが，本章で行なった数値計算においても非線形制御の適用の有効性が確認された．また，数値例においては学習の局所解が問題となったが，これは2章や3章で示した数値例ではあまり問題にならなかったので，本研究で用いたシミュレータにおいて制御系などに多くの整数型の変数を用いられていたことが局所解の原因になったと考えられる．

本章で行なった数値計算に用いた計算機の CPU は 800MHz 以上のものである．CPU の性能差による差が存在するものの，ロバスト性を考慮しない学習に要した時間は数十分程度であった．これに比べて，ロバスト性を考慮する方法における J_γおよび J_vによる学習方法では，統計量を計算する必要があるため，数十倍の演算が必要となる．最も多くの標本を用いる J_{max}により学習方法では，4.7.3節で示したアルゴリズムにより，短時間で学習を終了させることが可能であり，計算時間に多少のばらつきがあったが，数時間で学習が完了した．4.7節においてゲイン切換型 PD 制御系のパレート最適解を示したが，これらを求めるためには莫大な計算時間が必要であった．計算時間を短縮するために，メッシュあまり細かく設定せず，あらかじめ行なった粗い探索による情報に基づいて探索範囲を小さく限定した．小さく限定した探索範囲をさらに 6 台の計算機で分割して探索させたが，それぞれの計算機において最低 6 日間計算の計算が必要であった．このように，発見的手法によりフライトシミュレータを用いて制御系を設計するために必要な時間

は莫大であることが分かる．また，発見的な手法では構築することができる制御系は線形制御系あるいは，簡単なゲイン切換制御系に限定されるが，その制御性能はニューラルネットワークの学習により構築した制御系と比べるとかなり劣るものであった．計算時間，ロバスト性，制御性能のいずれをとっても，発見的な手法による制御系の構築法はニューラルネットワークの学習による制御系の構築法を上回ることはできなかった．また，それぞれの確率的な不確かさに対するロバスト制御系の学習では，それぞれの標本過程は独立・並行に行なうことが可能である．そこで複数の計算機をLANにより接続したPC Clusterシステムを構築し，計算処理速度を飛躍的に向上させることができることを示した．構築したPC Clusterシステムは確率的な不確かさに対するロバスト制御系の学習には不可欠なものであると考える．

本章において，制御対象とした無人ヘリコプタRMAXは，自律飛行が可能となることで応用範囲がさらに広がると考えられる．無人ヘリコプタが主として用いられている農薬散布などの自動化などが可能となり，農作業における省力化に貢献することが可能となることをはじめとして，危険地帯における観測活動や防災活動，遭難者の救難救助活動の支援としても利用できる．また，工場などの施設の監視システムや交通監視システムなど安全監視システムとしての価値も高い．より高度な目的を達成することができる自律型UAVを開発するためには，制御系の制御性能だけなく，信頼性も考慮して研究を進めていく必要がある．このためには，本章で示した学習方法により学習をおこなったニューラルネットワークをRMAXに搭載し飛行試験を行い，飛行実験データを得る予定である．また，4.4節で述べたように自律飛行制御系のためにはコマンドシステムなどの操縦インタフェース [57] や画像認識などによる状況判断 [49]，飛行経路の計画，航法システムの研究も重要であるので，今後も研究を引き続いて行なう必要があると考えている．

第 5 章

結言

本論文では, ニューラルネットワークの学習により制御系を構築する手法について考察を行うとともに, 提案方法が有効であることを無人機の自律飛行制御系の構築などに適用することにより示した. 各章において得られた結論を要約すると以下の通りである.

第2章では, ニューラルネットワークの学習により最適フィードバック制御系を構築する手法を示した。学習アルゴリズムとして, 勾配法に基づく学習アルゴリズムと Powell の共役方向法に基づく学習アルゴリズムの二つを提案し, それぞの特性を明らかにした. 特に, Powell の共役方向法に基づく学習アルゴリズムは勾配法に基づく学習アルゴリズムと比べると,

1. 微分不可能である関数を評価関数などに用いることが可能である.
2. 学習アルゴリズムに制御対象の情報を組み込む必要がないため, 汎用性が高く, 任意のシミュレータに組み込むことや, 実機を用いた実験からも学習することが可能である.

という利点を持つ. 前者の利点は最短時間制御系をはじめととして, 入力の切換を行なう制御系の学習の際に有効である. 実機を用いる実験から学習することは, 学習に必要である繰り返し数が多いため, 必要となる時間の問題となるので現実的ではないかもしれないが, 先験的な制御対象に関する情報を全く必要としないため, 任意のシミュレータにこの学習アルゴリズムを組み込むことが非常に容易である. しかし, ニューラルネットワークを制御系設計に応用する際の問題点は, 制御系の安定性である. 特に, 制御対象の正確なモデリングが困難であることが多く, 対象の数学モデルには一般的にモデル化誤差など不確かさを含むので, 制御対象に存在する不確かさに対してロバストに安定であることは, 制御系として望まれる性質の一つである. しかし, 本章で示した最適フィードバック制御系を学習する方法では, 最適制御理論からの類推からロバスト性が高いと期待するのみにとどまっており, 定量的にロバスト性の大きさを評価するこ

とができないことが問題である．このために，学習後のニューラルネットワークのロバスト性については，十分に検証を行なう必要がある．

第3章では，第2章の方法により学習を行なったニューラルネットワークのロバスト性は学習の際に予測することができず，期待されるロバスト性を持たないことがあることを示した．そこで本章では，このような難点を解消するために，ニューラルネットワークの学習によりロバスト制御系の構築を行なう方法を示した．まず，非構造的な不確かさを含む制御対象に対してロバスト制御系を構築するには，二つのニューラルネットワークの競合する学習が有効であることを提案した．さらに，ロバスト制御系でしばしば問題となる保守性を軽減するために，構造的な不確かさに対するロバスト制御系の学習方法を提案した．これらの方法では，学習の際にニューラルネットワークのロバスト性を定量的に設定することが可能であるが，この点が他のニューラルネットワークを用いた制御系の研究とは大きく異なる点であり，優位な点といえる．また，スライディングモード制御系の学習方法も提案し，制御入力の使用に厳しい制約が加わることにより，従来の方法では一致していた滑り面と入力切換面は一般には異なる面となることを示した．特に，滑り面を線形に設計した場合でも入力切換面は状態空間中で一般に曲がっていることが判明した．ニューラルネットワークの学習を用いることにより，従来法で困難であった非線形の入力切換面を設計や容易となる上に，活性化関数としてシグモイド関数を用いることにより，入力の連続化を目的とした境界層の設計も容易となる．また，本章で提案した方法を組み合わせることにより，複数の不確かさや部分的な情報を持つ不確かさに対するロバスト制御系の学習法や，ロバスト制御性能を定量的に評価することも可能となることを示した．更に，ニューラルネットワークの学習は一般の制御系のロバスト性の解析にも有効であり，任意の制御系のロバスト性の定量的な評価を行なうことが可能であることも示した．

第4章では，ニューラルネットワークの学習により，制御対象の情報を用いずに制御系を構築する方法を示し，無人ヘリコプタ RMAX のフライトシミュレータに学習アルゴリズムを組み込むことで自律飛行制御系を構築することによりその有効性を示した．制御対象の詳細な知識を必要としない本手法により制御系の開発を分散的に行なうことが可能となり，開発効率の向上に貢献することができる．また，確率的な不確かさに対してロバストな制御系をニューラルネットワークの学習により構築する方法を示し，その特性を明らかにするとともに，それらの組み合わせや併用により特性を改善する方法も示した．従来のロバスト制御系の構築法は，制御対象に関する十分な知識を必要としたが，提案手法により非常に簡単にロバスト制御系を構築することが可能となるので，ロバスト制御系を容易に用いることができるようになる考えられる．

製品の高度化は今後も進んでいくと思われるが，それに伴って高度なシステムの全体に関する詳細な知識を持つことはますます難しくなっていくであろう．また，システムの高度化にともない，制御系の開発も難しいものとなり，開発コストも大きなものとなると考えられる．高度なシステムでは．従来の方法により

制御系を構築することは難しくなり，経験と勘に依存する設計に頼ると制御系の性能や信頼性の確保が問題となるだろう．本研究による結果は，ニューラルネットワークをはじめとするインテリジェントシステムの制御系への応用により，このような問題を解決することが可能であるということ示している．高度なシステムの代表的な例といえる無人機では，その自律飛行制御系に対してインテリジェントシステムを応用することが試みられている．例えば，ファジー制御による飛行制御系の構築があるが，これはPD制御系のゲインをファジールールを用いて変更するものであった [58] が，高性能化・高信頼化が達成できたとはいえないと思われる．実際に，[58] ではファジー制御は姿勢制御系に対して目標指令を与えるのみであり，完全にファジー制御のみで無人機の飛行を行なうことはできないが，これはファジー制御系の構築の難しさが原因であると思われる．著者の知る限りでは完全にインテリジェントシステムのみを用いて，無人機の飛行制御に成功したという例はまだない．よって，飛行実験を行っていくことや，さらに研究を行って問題点を解決することで，完全にインテリジェントシステムのみによる無人機の飛行制御を完成させたいと考えている．また，無人機の自律飛行制御には，無人飛行であるがゆえに高い信頼性が不可欠である．よって，無人機のハードウェアの信頼性を向上させるだけでなく，ソフトウェア，すなわち，自律飛行制御系の信頼性の向上が必要である．このためには，ニューラルネットワークのオンライン学習による動的な補償やオフライン学習によるロバスト性の確保が有効で，現在の飛行制御系と比べると格段に信頼性の向上する．また，自律性を高めるためには，画像処理などにより飛行状況判断 [49] を行うことが必要となる．このように無人機の自律飛行制御系では第4章で述べたような階層的なアプローチが有効であるが，画像処理や意思決定においても，ニューラルネットワークなどのインテリジェントシステムが有効であることは多くの論文において提案されている通りである．今後も，インテリジェントシステムに関する研究を更に発展させていきたいと考えている．

謝辞

本研究を行なうにあたって，京都大学大学院工学研究科 井上紘一教授には親切な助言，親身なご指導をしていただきました．私が日本電気株式会社に在籍の間も，ニューラルネットワークの海外における研究動向などを細かにご教示してくださり，研究を続けるよう叱咤，激励をいただきました．さらに，京都大学にて研究を行なう機会を与えていただいたおかげで，本研究を遂行することができました．

京都大学大学院工学研究科 土屋和雄教授，吉川恒夫教授には本研究を最終的にまとめあげるために内容を精読していただいた上に，適切なご指摘と助言をいただきました．心より感謝するとともにお礼申し上げます．

また，京都大学大学院工学研究科 幸田武久助教授には，大学院在学中はもちろん，研究方法をはじめとして熱心にご指導していただきました．井上紘一教授，幸田武久助教授には，システム工学について幅広く最新の研究内容をご教示いただいただけでなく，数多く研究会に出席させていただき，多くの最新の研究成果に触れさせていただきました．これは大変貴重な経験でした．

京都大学大学院工学研究科航空宇宙工学専攻井上研究室 (振動制御工学分野) の先輩，同級生とは研究の進め方をはじめとして貴重な討論を行なうことができました．特に，教養学部時代からの友人たちには大変よい刺激を受け，特に計算機に関しては幅広い知識を得ることができました．このことが，航空工学教室において制御工学に関する研究を行うきっかけとなりました．また，井上研究室の助手として採用されてからは，多くの卒業生あるいは在学生と研究会や輪講，実験など貴重な討論を行なうことができました．4章の成果の一部は，橋本寛之君，細川尚美さん，吉永秀人君と討論を行ない研究を行なった結果として得られたものです．更に，4章の研究はヤマハ発動機株式会社スカイ事業部のご協力と共同研究する機会が与えられなければ行うことはできなかったと思います．特に，ヤマハ発動機株式会社 佐藤彰主事には無人ヘリコプタ RMAX に関する情報やフライトシミュレータのご提供をしていただきました．また，ヤマハ発動機株式会社スカイ事業部の皆様とは研究会を通じて，貴重な討論や情報提供をしていただきました．この場を借りてお礼申し上げます．

このように私一人だけで本研究を行うことは不可能であり，数多くの方のおかげで遂行することができました．最後に，両親や家族の協力がなければ，本研究は完結させることはできませんでした．心より感謝します．本当にありがとうございました．

中西 弘明 発表論文一覧

1. 中西 弘明, 幸田 武久, 井上 紘一: ニューラルネットワークによる最適フィードバック制御系の構成法, Proceedings of the 37th Annual Conference of the Institute of Systems, Control and Information Engineers, pp. 31-32, 1993

2. 中西 弘明, 幸田 武久, 井上 紘一: ニューラルネットワークによる最適フィードバック制御系の構成法, Proceedings of the 38th Annual Conference of the Institute of Systems, Control and Information Engineers, pp.141-142, 1994

3. Hiroaki Nakanishi, Takehisa Kohda and Koichi Inoue: A Design Method of Optimal Control System by Use of Neural Network, Proceedings of the 1997 International Conference on Neural Network, vol. 2, 871 – 875, 1997

4. 中西 弘明, 幸田 武久, 井上 紘一: ニューラルネットワークによる最適フィードバック制御系の構成法, -多目的計画法の応用- Proceedings of the 36th SICE Annual Conference, pp. 751 – 752, 1997

5. 中西 弘明, 幸田 武久, 井上 紘一: ニューラルネットワークによる最適フィードバック制御系の構成法, 計測自動制御学会論文集, vol. 33, no. 9, pp.882 – 889, 1997

6. 井上 紘一, 幸田 武久, 中西 弘明, 航空・宇宙機制御の基礎理論, 第 4 回:航空宇宙におけるシステム信頼性理論, 計測と制御, vol. 36, no. 11, pp. 818–825, 1997

7. 中西 弘明, 井上 紘一: ニューラルネットワークによるロバストな制御系の構築, Proceedings of the 42th Annual Conference of the Institute of Systems, Control and Information Engineers, pp. 93 – 94, 1998

8. Hiroaki Nakanishi and Koichi Inoue: Design Methods of Robust Feedback Controller by Use of Neural Networks, Proceedings of the International ICSC/IFAC Symposium on NEURAL COMPUTATION(NC'98), pp. 731 – 736, 1998

9. 中西 弘明, 井上 紘一: ニューラルネットワークを用いたロバスト制御系の学習, 第 8 回インテリジェント・システム・シンポジウム予稿集, pp. 523 – 528, 1998

10. 中西 弘明, 井上 紘一, 佐藤 彰: ニューラルネットワークの学習による産業用 RC ヘリコプターの自律制御系の設計法, Proceedings of the 43th Annual Conference of the Institute of Systems, Control and Information Engineers, pp. 401 – 402, 1999

11. Hiroaki Nakanishi and Koichi Inoue: Training Method for a Sliding Mode Controller and Quantified Robustness against Uncertainty, Proceedings of the 1999 International Joint Conference on Neural Network, 1999

12. 中西 弘明, 井上 紘一: ニューラルネットワークの学習によるロバストな制御系の構築法, システム制御情報学会論文誌, vol. 12, no. 10, pp. 625 – 632, 1999

13. Hiroaki Nakanishi, Akira Sato and Koichi Inoue: Design Method of an Autonomous Helicopter Flight Controller by a Neural Network, Proceeding of The 37th Aircraft Symposium, pp. 654 – 657, 1999

14. 中西 弘明, 井上 紘一: パラメータの不確かさに対するニューラルネットワークによるロバストな制御系の構築法, 第 9 回インテリジェント・システム・シンポジウム予稿集, pp. 229 – 234, 1999

15. 中西 弘明, 佐藤 彰, 井上 紘一: ニューラルネットワークの学習による無人ヘリコプターの自律飛行制御系の構築法, 平成 11 年度航空宇宙学会関西・中部支部合同秋季大会予稿集

16. Hiroaki Nakanishi and Koichi Inoue: Sliding Mode Controller by Use of Neural Networks, Elektrotecnik und Informationstechnik, Vol. 117, No. 1, pp. 36–42, Jan. 2000.

17. 橋本 寛之, 中西 弘明, 佐藤 彰, 井上 紘一: ニューラルネットワークの学習による無人ヘリコプターの自律飛行制御系の構築法, Proceedings of the 44th Annual Conference of the Institute of Systems, Control and Information Engineers, pp. 491 – 492, 2000

18. Hiroaki Nakanishi and Koichi Inoue: Design Methods of Robust Feedback Controller against Parametric Uncertainties by Use of Neural Networks, Proceedings of the International ICSC/IFAC Symposium on NEURAL COMPUTATION(NC'2000), 2000

19. Hiroaki Nakanishi and Koichi Inoue: Methods to Design Robust Controllers against Nonlinear and Multiple Uncertainties by Use of Neural Networks, Proceedings of the 2000 International Joint Conference on Neural Network, 2000

20. 中西 弘明 井上 紘一: ニューラルネットワークの学習によるロバスト制御性能を考慮した制御系の学習, 第 10 回 FAN インテリジェント・システム・シンポジュウム講演論文集 pp217–222, 2000

21. 中西 弘明 井上 紘一 佐藤 彰: 自律型エアロロボットの飛行制御へのニューラルネットワークの応用, 第 28 回知能システムシンポジウム資料 pp 203–208, 2001

22. 吉永 秀人 中西 弘明, 橋本 寛之, 佐藤 彰, 井上 紘一: ニューラルネットワークによる自律型エアロボットの制御系の構築法, Proceedings of the 45th Annual Conference of the Institute of Systems, Control and Information Engineers, pp. 83 – 84, 2001

23. Hiroaki Nakanishi, Akira Sato and Koichi Inoue: Development of Command and Control Systems for UAVs, Proceedings of AHS International Forum 57, 2001

24. Hiroaki Nakanishi and Koichi Inoue: Training method of robust controllers against stochastic disturbances, Proceedings of the 2001 International Joint Conference on Neural Network, 2001

25. 中西 弘明, 井上 紘一ニューラルネットワークの学習による確率的な外乱に対するロバスト制御系の構築法: 第 11 回 FAN インテリジェント・システム・シンポジュウム講演論文集 to be presented in next September at FAN'01, Osaka

26. 橋本 寛之, 吉永 秀人, 中西 弘明, 井上 紘一ニューラルネットワークよる UAV の自律飛行制御第 11 回 FAN インテリジェント・システム・シンポジュウム講演論文集 to be presented in next September at FAN'01, Osaka

27. 渡邊 洋子, 中西 弘明, 佐藤 彰, 井上 紘一: 無人ヘリコプタのためのボイスコマンドシステム構築, ヒューマンインタフェースシンポジウム 2001 講演論文集, 2001 (to be presented in next October at Osaka)

28. Hiroaki Nakanishi, Akira Sato, Takehisa Kohda and Koichi Inoue: to be presented in next November at APSS2001, Kyoto

29. 中西 弘明, 佐藤 彰, 井上 紘一ニューラルネットワークによる無人ヘリコプタの自律飛行制御の構築 SICE システム・情報部門学術講演会 2001 講演論文集 to be presented in next November at SICE'01, Miyazaki

30. 細川 尚実, 中西 弘明, 佐藤 彰, 井上 紘一自律型無人ヘリコプタの飛行制御系に関する検討平成 13 年度航空宇宙学会関西・中部支部合同秋季大会予稿集 to be presented in next November, Kyoto

参考文献

[1] K. Funahashi. On the Approximate Realization of Continuous Mappings by Neural Networks. *Neural Networks*, Vol. 2, No. 3, pp. 183–192, 1989.

[2] 丸山稔. Radial basis function を用いた学習ネットワーク. システム/制御/情報, Vol. 36, No. 5, pp. 322–329, 1992.

[3] D.E. Rumelhart, G.E. Hilton, and R.J. Wiliams. Learning Representations by Error Propagation. In *Parallel Distributed Processing*, Vol. 1, pp. 318–362. MIT Press, 1986.

[4] 西川, 北村編著. ニューラルネットと計測制御. 浅倉書店, 1995.

[5] D. Psaltis. A Multilayerd Neural Network Controller. *IEEE Control Systems Magazine*, pp. 17–21, Apr 1988.

[6] 山下, 島, 石動. ニューラルネットワークによる学習・適応制御. 計測と制御, Vol. 30, No. 4, pp. 302–307, 1991.

[7] 川人光男. 神経回路網と運動制御. ニューロコンピューティングの基礎理論, pp. 177–231, 1990.

[8] K.S. Narendra and K. Parthasarathy. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Tran. Neural Networks*, Vol. 1, No. 1, pp. 252–262, 1990.

[9] A.J. Calise and R.T. Rysdyk. Nonlinear adaptive flight control using neural networks. *IEEE Control Systems*, pp. 14–25, December 1998.

[10] B.S. Kim and A.J. Calise. Nonlinear Flight Control Using Neural Networks. *Journal of Guidance, Control and Dynamics*, Vol. 20, No. 1, pp. 26–33, 1997.

[11] Youji Iiguni, Hideaki Sakai, and Hidekatsu Tokumaru. A Nonlinear Regulater Design in the Presence of System Uncertainties Using Multilayered Neural Network. *IEEE Trans. Neural Networks*, Vol. 2, No. 4, pp. 410–417, 1991.

[12] K.S. Narendra. Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks. *IEEE Trans. Neural Networks*, Vol. 2, No. 4, pp. 252–262, 1991.

[13] D.H. Nyugen and B. Widrow. Neural Networks for Self-learnig Control Systems. *IEEE Control Systems Magazine*, pp. 18–23, 1990.

[14] 横山恭. 混合構造型ニューラルネットワークによるプラントの同定と最適制御入力のシンセシス. 京都大学工学研究科航空工学専攻修士論文, 1992.

[15] 平澤, 大林, 藤田, 古賀. 一般化学習ネットワーク理論. 電気学会論文誌 C 分冊, Vol. 116-C, pp. 794–801, 1996.

[16] 中西弘明. ニューラルネットワークを用いた最適フィードバック制御系の構成法. 京都大学工学研究科航空工学専攻修士論文, 1994.

[17] Yutaka Maeda and Rui J. P. De Figueiredo. Learning rules for neuro-controller via simultaneous perturbation. *IEEE Trans. Neural Networks*, Vol. 8, pp. 1119–1130, 1997.

[18] K.F. Fong and A.P. Loh. Discreate-Time Optimal Control Using Neural Nets. *1991 IEEE International Joint Article on Neural Networks (Cat. No.91CH3065-0)*, Vol. 2, pp. 1355–1360, 1991.

[19] T. Parisini and R. Zoppoli. Backpropagation for N-stage Optimal Control Problems. *1991 IEEE International Joint Article on Neural Networks (Cat. No.91CH3065-0)*, Vol. 2, pp. 1518–1529, 1991.

[20] A.E. Bryson. Jr. and Yu-Chi Ho. *Applied Optimal Control.* Revised Printing, 1975.

[21] F. L. Lewis and V. L. Syrmos. *Optimal Control(2nd. Edition).* John Wiley, 1995.

[22] 坂和愛幸. 最適化と最適制御. 森北出版, 1980.

[23] 辻輝生. 最適制御問題の数値解法と誤差範囲. 計測と制御, Vol. 25, No. 7, pp. 607–614, 1986.

[24] 濱田一男. ニューラルネットワークを用いた非線形最適レギュレータの構築. 京都大学工学研究科航空工学専攻修士論文, 1993.

[25] Z.V. Rekasius. Suboptimal Design of Intentionally Nonlinear Controller. *IEEE Trans. Automatic Control*, Vol. 11, No. 4, pp. 380–386, 1964.

[26] M.J.D. Powell. An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives. *Computer Journal*, Vol. 7, pp. 155–162, 1964.

[27] T.R. Niesler and J.J. du Plessis. Time-optimal control by means of neural network. *IEEE Control Systems*, Vol. 15, No. 5, pp. 23–33, 1995.

[28] M.W. Heiges,P.K. Menon and D.P. Schrage. Synthesis of a Helicopter Full Authority Controller. *Proceedings of the AIAA Guidance, Navigation and Control Article*, pp. 207–213, 1989.

[29] 馬場順昭, 高野博行. 非線形逆ダイナミクスを用いた飛行制御. 日本航空宇宙学会誌, Vol. 47, pp. 122–128, 1999.

[30] J. C. Doyle, K. Glover, P. P. Khargonekar and B. A. Francis. State-Space Solutions to Standard H_2 and H_∞ Control Problems. *IEEE Trans. Automatic Control*, Vol. 34, No. 8, pp. 831–847, 1989.

[31] 美多勉. H_∞制御. 昭晃堂, 1994.

[32] 中西弘明, 幸田武久, 井上紘一. ニューラルネットワークによる最適フィードバック制御系の構成法, -多目的計画法の応用-. *Proceedings of the 36th SICE Annual Article*, pp. 751–752, 1997.

[33] 中西弘明, 井上紘一. ニューラルネットワークを用いたロバスト制御系の学習. 第8回インテリジェント・システム・シンポジウム予稿集, pp. 523–528, 1998.

[34] 大林, 平澤, 須納瀬. 一般化ネットワークにおけるハイブリッド制御方式. 計測自動制御学会第36回学術講演会予稿集, Vol. 2, pp. 745–746, 1997.

[35] V. I. Utkin. Sliding Mode Control Design Principles and Applications to Electric Devices. *IEEE Trans. Industrial Electronics*, Vol. 40, No. 1, pp. 23–36, 1993.

[36] J. Y. Hung, W. Gao, and J. C. Hung. Variable Structure Control: A Survey. *IEEE Trans. Industrial Electronics*, Vol. 40, No. 1, pp. 2–22, 1993.

[37] 野波健蔵, 田宏奇. スライディングモード制御. コロナ社, 1994.

[38] 岡林亮爾, 古田勝久. 入力制限を考慮したスライディングモード制御系の設計. 計測自動制御学会論文集, Vol. 33, No. 12, pp. 1148–1154, 1997.

[39] D. J. Bell and D. H. Jacobson. *Singlar Optimal Control Problems.* Academic Press, 1975.

[40] 石島, 島, 石動他. 非線形システム制御論. 計測自動制御学会, 1993.

[41] 中西弘明, 井上紘一. ニューラルネットワークによる ロバストな制御系の構築. *Proceedings of the 42th Annual Article of the Institute of Systems, Control and Information Engineers*, pp. 93–94, 1998.

[42] 石動 他島. 非線形システム制御論. コロナ社, 1997.

[43] F. Glover. Tabu Search, Part 1. *ORSA Journal on Computation*, Vol. 1, No. 3, pp. 190–206, 1989.

[44] F. Glover. Tabu Search, Part 2. *ORSA Journal on Computation*, Vol. 2, No. 1, pp. 4–326, 1990.

[45] 鈴木弘人. 無人ヘリコプタ RMAX の概要. 第 37 回飛行機シンポジウム講演集, pp. 353–356, 1999.

[46] 佐藤彰. 無人ヘリコプタの自律飛行制御. 第 37 回飛行機シンポジウム講演集, pp. 349–352, 1999.

[47] 佐藤彰, 鈴木弘人, 山越隆夫. 無人ヘリコプタの自律飛行による有珠山火口付近の観測. 第 38 回飛行機シンポジウム講演集, pp. 109–112, 2000.

[48] Akira Sato. Research and Development and Civil Application of an Autonomous,Unmanned Helicopter. *Proceedings of AHS International Forum 57*, 2001.

[49] Omead Amidi. *An Autonomous Vision-Guided Helicopter.* PhD thesis, Carnegie Mellon University, 1996.

[50] Herbert E. Rauch. Intelligent fault diagnosis and control reconfiguration. *IEEE Control Systems*, pp. 6–12, June 1994.

[51] 金井喜美雄, 越智徳昌. Act によるフライトコントロール. 日本航空宇宙学会誌, Vol. 35, pp. 118–126, 1987.

[52] 橋本 寛之, 中西 弘明, 佐藤 彰, 井上 紘一. ニューラルネットワークの学習による無人ヘリコプターの自律飛行制御系の構築法. *Proceedings of the 44th Annual Article of the Institute of Systems, Control and Information Engineer*, pp. 491–492, 2000.

[53] P. Whittle. *Risk Sensitive Optimal Control.* John Wiley&Sons Ltd., 1990.

[54] 加藤寛一郎, 今永勇生. ヘリコプタ入門. 東京大学出版会, 1985.

[55] Gareth D. Padfield. *Helicopter Flight Dynamics:The Theory and Application of Flying Qualities and Simulation Modeling.* AIAA, 1996.

[56] 坂和正敏. 非線形システムの最適化<一目的から多目的へ>. 森北出版, 1986.

[57] 田村博. ヒューマンインタフェース. オーム社, 1998.

[58] 中村心哉, 菅野道夫. 画像情報を用いた無人ヘリコプタのトラッキング制御. 第12回ファジーシステムシンポジウム講演論文集, pp. 585–588, 1996.